Forth Weather Predictor

Dr. Dobb's Journal

For the Experienced in Microcomputing

#90 APRIL 1984
$2.95 (3.50 Canada)

A Public Key Cryptography System, Part II

Optimizing Strings in C

BASICFMT

Self-dFENSE™ for EDP managers.

The micro invasion has begun. And, chances are, you've now got a lot of different people in a lot of different departments using a lot of different micros.

Now there's a way for you to control and maximize the benefits of all the different micros in your domain.

Fight back with dBASE II.®

dBASE II is the relational database management system from Ashton-Tate that enables you to manage your micro-based corporate data resources with the high level of consistency and sophistication you've enjoyed with mainframe and minicomputer systems.

Armed with dBASE II and the dBASE II RunTime™ program development module, you can write programs which will enable micro users in each department to "do their own thing" while creating complete database consistency throughout the company.

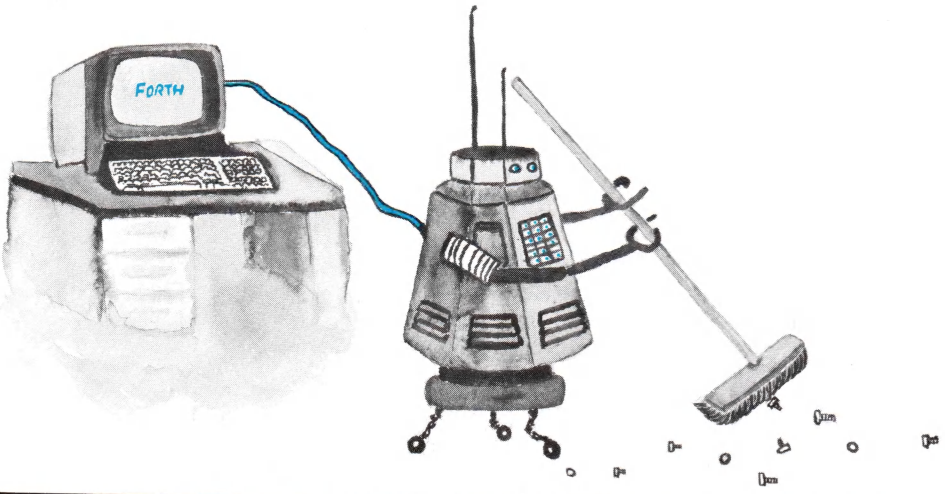
dBASE II is a powerful, flexible way for you to effectively manage the micro proliferation.

Help is here.

If you'd like to know more about how dBASE II and RunTime can help you win the micro management battle, contact Ashton-Tate today. 10150 West Jefferson Boulevard, Culver City, CA 90230. (800) 437-4329, ext. 217 . In Colorado (303) 799-4900. In the U.K. (0908) 568866.

ASHTON·TATE™

FORTH GIVES YOU TOTAL CONTROL



GRAPHICS • GAMES • COMMUNICATIONS ROBOTICS • DATA ACQUISITION • PROCESS CONTROL

***FORTH:* for Z-80®, 8080, 8086, 68000, and IBM® PC**
(Complies with the New 83-Standard)

- ***FORTH* programs are instantly portable across the four most popular microprocessors.**
- ***FORTH* is interactive and 20 times faster than BASIC.**
- ***FORTH* programs are highly structured and easy to maintain.**
- ***FORTH* provides direct control over all interrupts, memory locations, and i/o ports.**
- ***FORTH* allows full access to DOS files and functions.**
- ***FORTH* application programs can be distributed as turnkey COM files with no license fee.**
- ***FORTH* Cross Compilers are available for ROM'ed or disk based applications on most microprocessors.**
- **Custom programming, consulting, and educational services available.**

FORTH Application Development Systems include interpreter/compiler with virtual memory management and multitasking, assembler, full screen editor, decompiler, utilities, and detailed technical manual. Standard random access files used for screen storage, extensions provided for access to all operating system functions. **Z-80 FORTH** for CP/M® 2.2 or MP/M II, $50.00; **8080 FORTH** for CP/M 2.2 or MP/M II, $50.00; **8086 FORTH** for CP/M-86 or MS-DOS, $100.00; **PC/FORTH™** for PC-DOS, CP/M-86, or CCPM, $100.00; **68000 FORTH** for CP/M-68K, $250.00

FORTH + Systems are 32 bit implementations that allow creation of programs as large as 1 megabyte. The entire memory address space of the 68000 or 8086/88 is supported directly for programs and data.

PC FORTH +	$250.00
8086 FORTH + for CP/M-86	$250.00
68000 FORTH + for CP/M-68K	$400.00

FORTH Cross Compiler allows you to customize the FORTH nucleus, recompile on a host computer for a different target computer, generate headerless and ROM-able code. Supports forward referencing. Produces executable image in RAM or disk file. No license fee for applications, $300.00.

FORTH Native Code Compilers

For Z80 FORTH and CP/M	$100.00
For 8086 FORTH and CP/M-86	$200.00
For IBM PC and PC-DOS	$200.00

Extension Packages
Software floating point (Z-80, 8086, PC only), $100.00; AMD 9511 support (Z-80, 8086, 68000 only), $100.00; Intel 8087 support (8086, PC only), $100.00; Advanced color graphics (PC only), $100.00; Symbolic interactive debugger (PC only), $100.00; PC/TERM Communications/file transfer for Smartmodem, $60.00; Cross reference utility, $25.00; PC/GEN (custom character sets, PC only), $50.00; Curry FORTH Programming Aids, $150.00; B-Tree index manager, $125.00; B-Tree index and file manager, $200.00; QTF + Screen editor for IBM PC, $100.00; Quad Integer Math Pack, $25.00.

AUGUSTA, Ada subset compiler from Computer Linguistics for Z-80 CP/M 2.2 systems, $90.00

"Starting FORTH" tutorial by Brodie, softcover, $16.00.

INTEL 8087-3 Numeric Coprocessor, $250.00

Z-80 and 8080 *FORTH* require 48 Kbytes RAM. 8086 and 68000 *FORTH* require 64 Kbytes RAM. Disk formats available include: 8" standard CP/M SSSD, Northstar 5¼" QD, Kaypro 5¼", Apple 5¼", Micro-Mate 5¼", MS-DOS 5¼", Osborne 5¼", DD, and Sage. Most other formats can be special ordered. Dealer inquiries invited.

Z-80 is a registered trademark of Zilog, Inc.; CP/M is a registered trademark of Digital Research, Inc.; IBM is a registered trademark of International Business Machines Corp.; Augusta is a trademark of Computer Linguistics; PC/FORTH and PC/GEN are trademarks of Laboratory Microsystems Inc.

LMI
Laboratory Microsystems Incorporated
4147 Beethoven Street, Los Angeles, CA 90066
*Phone credit card orders to **(213) 306-7412***

VISA® MasterCard

Circle **no. 29** on reader service card.

From the Rainbow to the VAX . . . from the PC Jr. to the 370.

Whitesmiths' C and Pascal Compilers continue to set the industry standards

- **FLEXIBILITY** — the same compiler
- **PORTABILITY** — dozens of systems
- **PORTABLE C LIBRARY** — UNIX compatible
- **COMPATIBILITY** — new UNIX-style library
- **AFFORDABILITY** — high portability
- **EASY LICENSING** — the Authorization Seal
- **RELIABILITY** — thousands of installations
- **COMPLETENESS** — the full language

DISTRIBUTOR AND DEALER INQUIRIES INVITED

Host:	Cross To: CP/M-80	CP/M-68K	MS/PC-DOS CP/M-86	P/OS
UNIX				
IDRIS/R11	✓	✓	✓	
RSX-11	✓	✓	✓	✓
RT-11	✓	✓	✓	
VMS-VAX	✓	✓	✓	✓
UNIX-4.1 BSD	✓	✓	✓	
UNIPLUS	✓		✓	
IDRIS/S68K	✓		✓	
MS/PC-DOS				
CP/M-86	✓	✓		
IBM 370			✓	

Host Only: CP/M-80* CP/M-68K P/OS-11 VERSAdos
*2.1 Cross Compilers available

BUILD A STRONG FOUNDATION WITH THE COMPILERS OF THE 80's

Whitesmiths, Ltd.
97 Lowell Road
Concord, MA 01742 (617) 369-8499
TLX 951708 SOFTWARE CNCM

Distributors: Australia, Fawnray Pty. Ltd., Hurstville, (612) 570-6100; Japan, Advanced Data Controls Corp., Chiyoda-ku, Tokyo (03) 263-0383; United Kingdom, Real Time Systems, Douglas, Isle of Man 0624833403; Sweden, Unisoft A.B., Goteborg, 031-13-56-32.

Dr. Dobb's Journal

For the Experienced in Microcomputing

April 1984
Volume 9, Issue 4

CONTENTS

ARTICLES

18 **Optimizing Strings in C**
by Edward McDermott
The decision to provide new capabilities to one's compiler should be made carefully. Finding that C had no string-manipulation verbs, the author decided to optimize his compiler's string handling. In addition to his routines, this article examines some of the considerations involved in such a decision. **(Reader Ballot No. 192)**

24 **Expert Systems and the Weather**
by Jack Park
The author chose predicting the weather as the subject of an early effort at knowledge engineering using EXPERT-2, a tiny, Forth-based, fifth-generation type language. With it, users can write high-level programs for performing logic inferences using rules written into their programs. The weather predictor presented here illustrates how a set of rules can be implemented or customized. **(Reader Ballot No. 193)**

32 **RSA: A Public Key Cryptography System, Part II**
by C. E. Burton
In Part I of this article, the mathematical foundations of a Rivest-Shamir-Adleman public key encryption system were explored. This final installment discusses the advantages of a PKS, as well as the generation of the keys, encryption and decryption of text, and the use of "digital signatures" to assure authenticity of a message's origin. **(Reader Ballot No. 194)**

60 **BASICFMT for TRS-80 (A BASIC Formatter)**
by Davy Crockett
No need to save a separate ASCII version of your BASIC program. This routine reads a BASIC program in compressed format from disk and prints a readable listing. **(Reader Ballot No. 195)**

DEPARTMENTS

6 Editorial

8 Letters

10 Dr. Dobb's Clinic
Super Directory Fix; RAM-Drive Needed; The Great American Merchandiser; What's So Hard About CP/M Plus?; CP/M Plus, Apple and ALS; How It's Done; Our April Item **(Reader Ballot No. 190)**

12 CP/M Exchange
CP/M V2.2 Compatibility; BDOS Function 10: Vastly Improved **(Reader Ballot No. 191)**

76 Software Reviews
microSUB:MATH; METAFILE Application Development System

86 16-Bit Software Toolbox
More on MS-DOS EXEC Function; PC-DOS Manual Typo; IBM PC Junior; MS-DOS Volume Labels; Finding Size of TPA under MS-DOS 2.0; CP/M Discrepancies **(Reader Ballot No. 196)**

94 C/Unix Programmer's Notebook
Comments about Unix; Low Level Input-Output in C **(Reader Ballot No. 197)**

98 Of Interest
(Reader Ballot No. 198)

102 Advertiser Index

Publisher - Jane Nissen Laidley

Editorial
Editor – Reynold Wiggins
Managing Editor – Randy Sutherland
Contributing Editors –
Robert Blum, Dave Cortesi, Ray Duncan, Anthony Skjellum, Michael Wiesenberg

Marketing
Marketing Director – Beatrice Blatteis
Marketing Assistant – Sally Brenton
Marketing Coordinator – Ed Gueble

Advertising
Advertising Director – Beatrice Blatteis
Advertising Sales –
Alice Hinton, Walter Andrzejewski

Circulation
Circulation Director - Terri Pond
Circulation Assistant - Frank Trujillo

Production
Art Director/Production Manager –
Shelley Rae Doeden
Production Coordinator – Alida Hinton
Typesetter – Jean Aring

Cover Illustration – Inge Infante

Subscription Rates: $25 per year within the United States; $44 for first class to Canada and Mexico; $62 for airmail to other countries. Payment must be in U.S. Dollars, drawn on a U.S. Bank.

Donating Subscribers Contributing Subscriber: $50/year ($25 tax deductible). **Retaining Subscriber:** $75/year ($50 tax deductible). **Sustaining Subscriber:** $100/year ($75 tax deductible). **Lifetime Subscriber:** $1000 ($800 tax deductible). **Corporate Subscriber:** $500/year ($400 tax deductible, recieves *five* one-year subscription).

Contributing Subscribers: Christine Bell, W. D. Rausch, DeWitt S. Brown, Burks A. Smith, Robert C. Luckey, Transdata Corp., Mark Ketter, Friden Mailing Equipment, Frank Lawyer, Rodney Black, Kenneth Drexler, Real Paquin, Ed Malin, John Saylor Jr., Ted A. Reuss III, Info World, Stan Veit, Western Material Control, S. P. Kennedy, John Hatch, Richard Jorgensen, John Boak, Bill Spees, R. B. Sutton, Stan Veit. **Lifetime Subscribers:** Michael S. Zick, F. Kirk.

Writer's Guidelines: All items should be typed, double-spaced on white paper. Listings should be produced by the computer, using a *fresh, dark ribbon* on continuous white paper. Please avoid printing on perforations. Requests to review galleys must accompany the manuscript when it is first submitted. Authors may receive a copy of the complete writer's guidelines and the current compensation schedule by sending a self-addressed, stamped envelope.

Foreign Distributors UK & Europe: Homecomputer Vertriebs HMBH 282, Flugelstr. 47,4000 Dusseldorf 1, West Germany; Euro Computer Shop, 182 Rue du Faubourg St. Denis, 75010, Paris, France; La Nacelle Bookstore, Procedure D'Abonnement 1-74, 2, Rue Campagne – premiere, F-75014, Paris, France; Computercollectief, Amstel 312A, 1017 AP Amsterdam, Netherlands. **Asia & Australia:** AXCII Publishing, Inc., 4F Segawa Bldg. 5-2-2, Jingumae, Shibuya-Ku, Tokyo 150, Japan; Computer Services, P.O. Box 13, Clayfield QLD 4011, Australia; Computer Store, P.O. Box 31-261, 22B Milford Rd., Milford, Auckland 9, New Zealand. *(Write for Canadian Distributors)*

EDITORIAL

Dr. Dobb's Journal began as a forum for sharing information and ideas about programming and computers. It continues to be a place to present new languages, utilities, tools, applications, algorithms, discoveries, and techniques to the microcomputing community. That community may then use, comment upon, improve, and expand these items. Many of the things explored within our pages have had major influence in the industry. *DDJ* readers have always been at the forefront of the microcomputer industry, providing a wealth of expertise to fuel discussion. Our authors have come primarily from within the readership, and it is this reader involvement that has sustained and guided the *Journal* through the years.

The material currently discussed in our pages is far more complex than it was eight years ago, and this has forced some adjustments. We continue to print complete listings of useful and innovative software, and to explore developments in areas such as languages and operating systems. Maintaining the community discussion of these larger, more complex ideas, however, has become increasingly cumbersome. We have periodically instituted reader-driven columns to provide more intimate exchange in areas of particular reader interest, the latest addition dealing with C and Unix. Most recently, we formed a board of referees (again, from within the readership) to help ensure that the articles published are as appropriate and accurate as possible.

From our homespun beginnings with tiny BASIC and tiny computers, we have grown with the microcomputing industry and community. Cosmetic changes have occurred to improve appearance and marketability. Fiscal considerations have prompted us to accept advertising, though we still work to keep the percentage down and to maintain our objectivity. And it was advertising revenue that helped us to begin paying authors within the last few months. Each move has been made because we felt it would allow us to better perform our function without compromising our traditions.

Now we have had a new opportunity presented to us – one which can open the door for many of the projects that we have longed to pursue, and one which will provide added resources with which to continue improving the quality of material we bring to you. This new opportunity will allow the tradition of *DDJ* to continue and allow exploration of new vistas.

We are joining forces with M&T Publishing, Inc., a company dedicated to quality and integrity, a company which likes what *DDJ* has been doing and would like to see it done even better. Laird Foshay, the president of M&T Publishing, was part of the *DDJ* staff in the past, and has always been interested in our objectives. People's Computer Company will still own *DDJ* while M&T will be licensed to publish the magazine. PCC's Executive Director will chair an advisory board set up to ensure that *DDJ* maintains its focus and direction. The current staff, slightly expanded, will continue to produce the magazine as usual.

Among the new faces will be Michael Swaine, who is joining the staff as editor-in-chief. Many of you may recognize the name from his three-year tour at *InfoWorld* as a senior editor. A long-time reader of *DDJ*, Mike is dedicated to the traditions of the *Journal.* A professional journalist and computer enthusiast, Mike is interested in continuing to improve the quality and scope of the service we provide. We are excited to have him on the team.

So what does this all mean? Well, it means that we will have a new address. It means new projects (perhaps a bulletin board here at the *DDJ* office) and new opportunities. It means better service to our subscribers and better communication links to our readers. It means more quality, more information, more of what you buy *DDJ* for. We will continue to explore ways to facilitate the dialogue among the microcomputing community. Don't look for radical changes in the *Journal;* we have no intention of diverting from our long-chosen path. Expect instead a continuation of our gentle evolution.

Reynold Wiggins

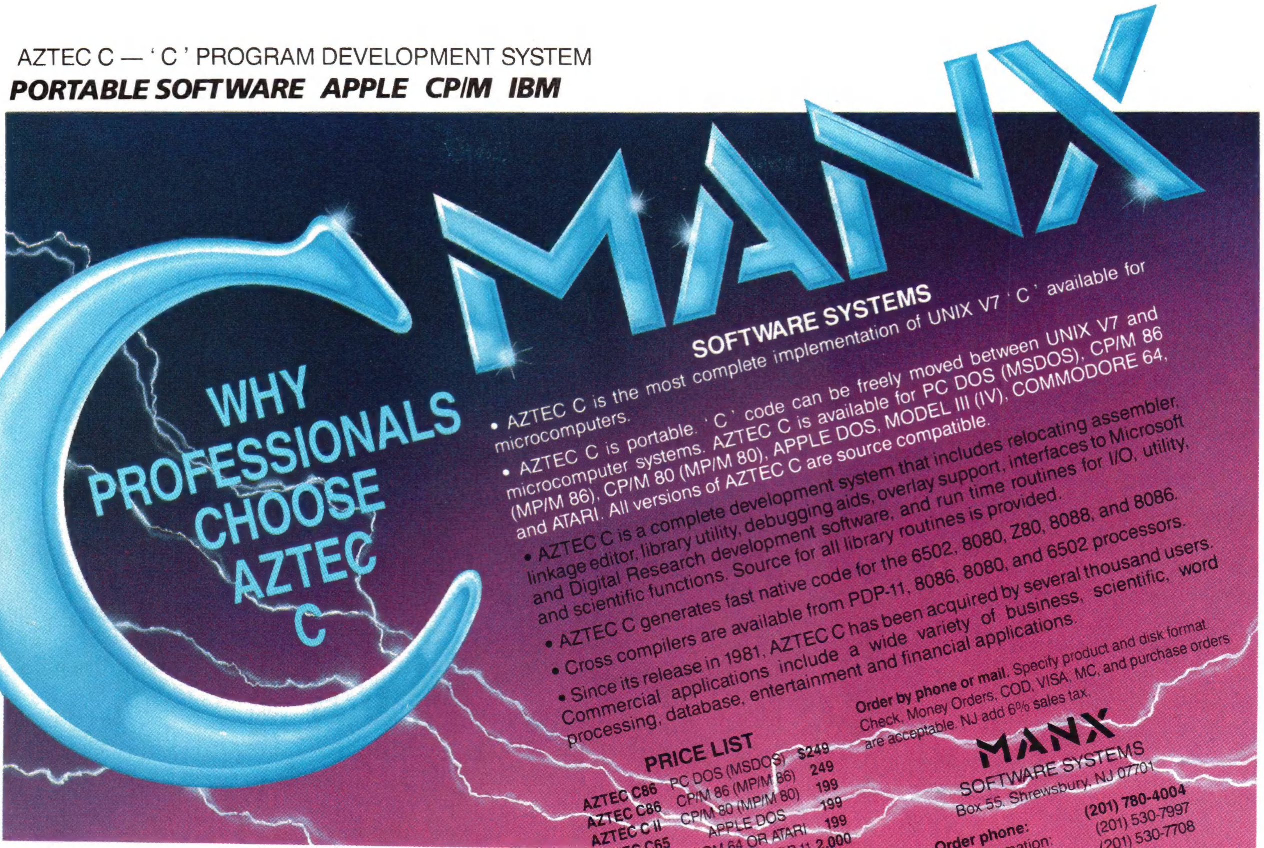
THIS MONTH'S REFEREES

A. Gomez, Telecomp, Inc.
Kim Harris, Forthright Enterprises
Richard Relph
John K. Taber, IBM
Bob Wheeler, Custom Hardware

Dr. Dobb's Journal (USPS 307690) is published twelve times per year by People's Computer Company, P.O. Box E, Menlo Park, CA 94026. Second class postage paid at Menlo Park, California 94026 and additional entry points. Address correction requested. Postmaster: send form 3579 to People's Computer Company, Box E, Menlo Park, CA 94026 · 415/323-3111 **ISSN 0278-6508**

AZTEC C — ' C ' PROGRAM DEVELOPMENT SYSTEM

PORTABLE SOFTWARE APPLE CP/M IBM



SOFTWARE SYSTEMS

- AZTEC C is the most complete implementation of UNIX V7 ' C ' available for microcomputers.
- AZTEC C is portable. ' C ' code can be freely moved between UNIX V7 and microcomputer systems. AZTEC C is available for PC DOS (MSDOS), CP/M 86 (MP/M 86), CP/M 80 (MP/M 80), APPLE DOS, MODEL III (IV), COMMODORE 64, and ATARI. All versions of AZTEC C are source compatible.
- AZTEC C is a complete development system that includes relocating assembler, linkage editor, library utility, debugging aids, overlay support, interfaces to Microsoft and Digital Research development software, and run time routines for I/O, utility, and scientific functions. Source for all library routines is provided.
- AZTEC C generates fast native code for the 6502, 8080, Z80, 8088, and 8086.
- Cross compilers are available from PDP-11, 8086, 8080, and 6502 processors.
- Since its release in 1981, AZTEC C has been acquired by several thousand users. Commercial applications include a wide variety of business, scientific, word processing, database, entertainment and financial applications.

PRICE LIST

AZTEC C86	PC DOS (MSDOS)	**$249**
AZTEC C86	CP/M 86 (MP/M 86)	**249**
AZTEC C II	CP/M 80 (MP/M 80)	**199**
AZTEC C65	APPLE DOS	**199**
AZTEC C65	COM 64 OR ATARI	**199**
AZTEC Cross Compilers	PDP-11	**2,000**
AZTEC Cross Compilers	Other	**500**

Order by phone or mail. Specify product and disk format. Check, Money Orders, COD, VISA, MC, and purchase orders are acceptable. NJ add 6% sales tax.

MANX SOFTWARE SYSTEMS
Box 55, Shrewsbury, NJ 07701

Order phone:	**(201) 780-4004**
Tech information:	(201) 530-7997
Tech support:	(201) 530-7708

Shipping: COD, 2nd day delivery, or Canada, add $5. Canada 2nd day or US next day delivery, add $20. Outside North America, add $20, and for 2nd day add $75.

Circle **no. 32** on reader service card.

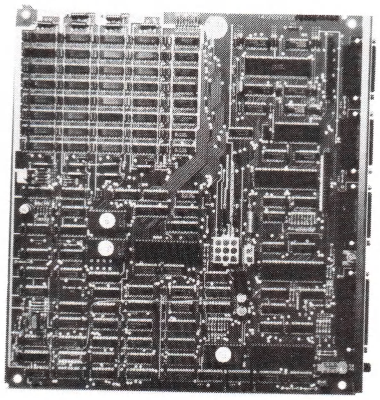
Z80* SINGLE BOARD COMPUTER!

64K RAM — 80 x 24 VIDEO DISPLAY — FLOPPY DISK CONTROLLER
RUNS CP/M* 2.2!

BRAND NEW PC BOARDS WITH SCHEMATICS!

BOARD MEASURES 11½" x 12½"

ALL ORDERS WILL BE PROCESSED ON A *STRICT*, FIRST COME, FIRST SERVED BASIS! ORDER EARLY!



$29.95
(BLANK BOARD WITH DATA AND ROM'S.)

NEW PRICE

GROUP SPECIAL: BUY 6 FOR $165!

USES EASY TO GET PARTS!

UNBELIEVABLE LOW PRICE!!! GIANT COMPUTER MANUFACTURER'S SURPLUS!

Recently Xerox Corp. changed designs on their popular 820* computer. These prime, new, 820-1 PC boards were declared as surplus and sold. Their loss is your gain! These boards are 4 layers for lower noise, are solder masked, and have a silk screened component legend. They are absolutely some of the best quality PC boards we have seen, and all have passed final vendor QC. Please note, however, these surplus boards were sold by Xerox to us on an *AS IS* basis and they will not warranty nor support this part.

We provide complete schematics, ROM'S, and parts lists. If you are an *EXPERIENCED* computer hacker, this board is for you! Remember, these are prime, unused PC boards! But since we have no control over the quality of parts used to populate the blank board, we must sell these boards as is, without warranty. You will have to do any debugging, if necessary, yourself!

*CP/M TM OF DIGITAL RESEARCH INC. (CALIF.) 820 TM OF XEROX CORP. Z80 TM OF ZILOG

WE ALSO CARRY LS, Z-80, EPROM'S, ETC. SEND FOR FREE CATALOG!

ADD $2 PER PC BOARD FOR SHIPPING. (USA and Canada)

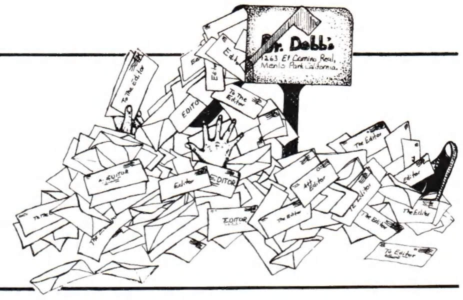
B. G. MICRO

P. O. Box 280298 Dallas, Texas 75228
(214) 271-5546

TERMS: Orders over $50 add 85¢ insurance. No COD. Tex. Res. Add 6% Sales Tax. Subject to prior sale. Foreign orders: US funds only. We cannot ship to Mexico. Foreign countries other than Canada add $6 per board shipping.

Circle **no. 5** on reader service card.

LETTERS



NBASIC, Postprocessed

Mr. Shammas' NBASIC program should not be confused with the NBASIC program marketed by the Computer Toolbox, Inc. – Ed.

Dear Sirs:

In reference to the article about NBASIC on page 24 of your 1/84 issue, I suspect the program has the following errors:

(1) Line 1220. If a variable has 'REM' in it and it appears on a line before a Remark, it will be changed to a single quote.

(2) Line 1120. If a filename is entered without a period in it, then line 1120 never gets executed and the output filename FOUT $ never gets defined.

I have several preprocessors and use them a lot. I think they should be promoted more in all computer journals.

Sincerely yours,
Peter Ansbacher
918 South Rustic Road
Columbia, MO 65201

Dear *DDJ*:

I read the article "NBASIC: A Structured Preprocessor for MBASIC" in the January 1984 issue. In running the program on a test sample of code I discovered a problem with the routine for converting remark statements to single quote marks. On page 27 the following lines are given for the conversion:

```
1210 FOR I=1 TO N
1220        P=INSTR (L$ (I), "REM")
1230        IF P<>0 THEN L$ (I)=
            MID$ (L$ (I) ,1,P-1)+
            "'" + MID$ (L$ (I) ,P+3)
1240 NEXT I
```

This will replace the first occurrence of the string "REM" in each line of text. However, the character may appear within quoted strings or as part of a variable name. It will cause execution problems if the variables and strings are changed. Mr. Shammas' program will change the test code in Listing One (at right) into the text in Listing Two (at right).

Making a few changes to his code will intelligently change the remark statements. The speed for the conversion is about the same as Mr. Shammas'. Delete lines 1210 and 1220 and add the lines in Listing Three (at right) to the program. Listing Four (at right) shows the output from the corrected program. This should handle most programs written in MBASIC.

This code may be improved upon, however it does work. Thank you for such a fine journal.

Sincerely,
Keith L. Terrill
603 Court Street
Syracuse, New York 13208

Listing One

```
10  REM
20  REM              THIS IS A TEST FOR THE REM PROGRAM
30  REM
40  A=1  :  B=2  :  C=3  :  REM THIS IS ANOTHER REM LINE
50  INPUT "THIS REM SHOULD NOT BE CHANGED";A
60  PRINT "THIS REM SHOULD REM BE CHANGED"  :
         REM BUT THIS ONE SHOULD
70  DATA ONE, 1, TWO, 2, REM, REM1, REM2
80  END
```

Listing Two

```
10  '
20  '                THIS IS A TEST FOR THE REM PROGRAM
30  '
40  A=1  :  B=2  :  C=3  :  ' THIS IS ANOTHER REM LINE
50  INPUT "THIS  ' SHOULD NOT BE CHANGED"; A
60  PRINT "THIS  ' SHOULD REM BE CHANGED"  :
         REM BUT THIS ONE SHOULD
70  DATA ONE, 1, TWO, 2,  '  , REM1, REM2
80  END
```

Listing Three

```
1209   QT$=CHR$ (34)
1210   FOR I=1 TO N
1213        P=INSTR (L$ (I) ,"REM") : IF P=0 THEN 1240
1216        P1=INSTR (L$ (I) , "DATA") : IF P1=0 THEN 1222
1219        IF P>P1 THEN 1240
1222        P2=1
1225        P1=INSTR (P2, L$ (I) ,QT$) : IF P1>P OR P1=0 THEN 1237
1228        P2=INSTR (P1+1,L$ (I),QTS) : IF P2=0 THEN 1240
1231        P2=P2+1 : IF P2<P THEN 1225
1234        P=INSTR  (P2,L$ (I) ,"REM") : IF P=0 THEN 1240 ELSE 1225
1237        L$ (I)=MID$ (L$ (I) ,1,P-1)+"'"+MID$ (L$ (I) ,P+3)
1240   NEXT I
```

Listing Four

```
10    '
20    '    THIS IS A TEST FOR THE REM PROGRAM
30    '
40    A=1 : B=2 :  C=3 : 'THIS IS ANOTHER REM LINE
50    INPUT "THIS REM SHOULD NOT BE CHANGED"; A
60    PRINT "THIS REM SHOULD REM BE CHANGED" :
          'BUT THIS ONE SHOULD
70    DATA ONE, 1, TWO, 2, REM, REM1, REM2
80    END
```

Fussin' over Sal/80

Dear Doctor:

This is to express my heartfelt appreciation for the very thorough and objective review of our SAL/80 package in the February issue by Jim Kronman. I'm embarrassed to take *any* exceptions to what was written, *but* . . .

One thing I do feel obliged to remark on: Jim's example of code-density compares two printer drivers, one in naked assembler and one in SAL/80. The SAL/80 driver is 400 bytes larger than the naked assembler, which might seem to imply that SAL/80 has a run-time package of 400 bytes. This is *not* the case. SAL/80 has *no* run-time package (and *no* royalties for commercial distribution of code generated using SAL/80).

But then it might seem to imply that the code emitted by SAL/80 is 40% larger than that obtained by coding in naked assembler, which is even *worse*! I haven't seen the two drivers in question, but I am quite certain that there are differences between the two which would account for all except a maximum of about 100 bytes plus perhaps 5% of the remaining difference.

There are several UTILITIES in the packages which, all taken together, might contribute perhaps 100 bytes of excess code because of their generality. And the test/branch code can typically be squeezed by about 5% in the average case. The 5% results from the fact that jumps to jumps do occur when a forward branch is emitted in the last group of a loop-body (hence, three bytes may be squeezable), and that the HL register is saved in certain memory-reference comparisons (which gives two possibly squeezable bytes if HL was inactive at the time).

The reason I'm making such a fuss about this is that SAL/80 is the only structured assembly package I know of that actually does *optimize* the test/branch code! There is a huge decision-tree in the compiler which does all the tricks dear to the hearts of assembly-language programmers to avoid writing two conditional jumps to implement LE (<=) and GT (>) comparisions. On a branch-by-branch basis, SAL/80 emits the tightest code possible if the double-registers are to be preserved over the comparison.

Cordially,
Steve Newberry
Protools
24225 Summerhill Avenue
Los Altos, CA 94022

Ackermann Uncovered

Dear Editor,

In response to Paul Condon's question (*DDJ* No. 88, February 1984, p. 9) "Who is Ackermann and where did he publish a definition of this function?" the following:

Wilhelm Ackermann died about twenty years ago. A student of David Hilbert's, he was one of the giants in research in the Foundations of Mathematics movement of the early part of this century. The function now associated with his name first appeared in a research monograph entitled "Zum Hilbertschen Aufbau der reelen Zalen" ("On Hilbert's Reconstruction of the Real Numbers") in *Math. Annalen,* Vol. 99 (1928), pp. 118-133.

The function was used to disprove (by counter-example) a conjecture by Hilbert to the effect that the set of real numbers definable by infinite sequences of what are now called "primitive-recursive" functions were, in an intuitive sense, all the real numbers that exist. Part of the argument of supposing this to be true was another conjecture that all recursive functions are primitive-recursive. The Ackermann function is *not* a primitive-recursive function, because it can be shown to "grow faster" than any primitive-recursive function. So, by showing that there are recursive functions which are not primitive-recursive, Ackermann disproved the first conjecture. Later, in the '30s, it was shown by Alan Turing that there are real numbers which are not definable even in terms of the most broadly generalized form of recursive function.

The original formulation of Ackermann's was slightly more complex than that now used, which is:

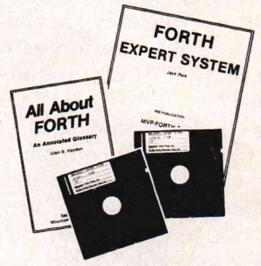
$$\begin{aligned} \mathrm{Ack}(0, n) &= n+1 \\ \mathrm{Ack}(m+1, 0) &= \mathrm{Ack}(m, 1) \\ \mathrm{Ack}(m+1, n+1) &= \mathrm{Ack}(m, \mathrm{Ack}(m+1,n)) \end{aligned}$$

A fuller discussion will be found in Hans Hermes' *Enumerability, Decidability, Computability,* Academic Press (1965), p. 84, and in Rozsa Peter's *Recursive Functions*, 3rd Revised Edition, Academic Press (1967), p. 106.

Didactically,
Steve Newberry
Newberry Microsystems
24225 Summerhill Ave.
Los Altos, CA 94022

DDJ

Learn Expert Programming



CREATE EXPERT SYSTEMS

Now, you can learn and create Expert or "knowledge-based" computer systems. Move into the 5th generation of computer programming with the **MVP-FORTH EXPERT TOOLKIT.** With this simple inference program you can learn and then create and run your own Expert programs. Examples are included of Stock Market Analysis, Animal Game and Digital Fault Analyzer.

The **MVP-FORTH EXPERT TOOLKIT** contains:

- Manual, by Jack Park, with tutorial on Expert Systems & Knowledge Engineering, Examples, bibliography and source code
- "All About FORTH" by Haydon, MVP-FORTH Glossary Disks with MVP-FORTH and EXPERT-2

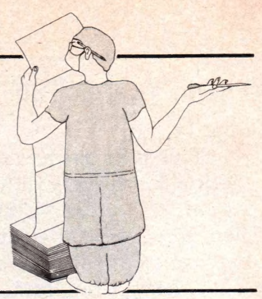
$100.00 (Calif. Residents add sales tax)

Specify: ☐APPLE II series ☐IBM PC or ☐8″ CP/M

MOUNTAIN VIEW PRESS, INC.
P.O. Box 4656
Mountain View, CA 94040
415/961-4103

Circle **no. 37** on reader service card.

DR. DOBB'S CLINIC



by D.E. Cortesi, Resident Intern

Super Directory Fix

A while back we told how David McLanahan found that the public-domain program SD (one of the extended DIR programs) failed under CP/M Plus. Ted Shapin, of Orange, CA, writes to say that "The SD program can easily be fixed to give the correct free space under CP/M 3.0. I fixed my copy with information from D. Sternlight. At the label TMOVE there is an instruction, LDAX D. A few lines further there is a second LDAX D. After this instruction insert the line

```
ANI OFH ;DROP ARCHIVE FLAGS
```

Meanwhile, McLanahan had found another solution. A later version of SD, SD48, works just fine on CP/M Plus.

RAM-Drive Needed

Malcom Fordham, of Bowie, MD, has "an IBM PC on which I use PL/I-86 under CP/M-86. I have sufficient memory to run the compiler and linker from a RAM disk. However, so far I haven't been able to find the software to do this under CP/M-86. Do you know of such a program?"

We only know that Concurrent CP/M for the PC contains RAM-drive support out of the box. Does anyone know of RAM-drive support for plain old CP/M-86? Maybe one of the makers of add-on RAM cards supplies one; they usually supply the support code for PC-DOS.

The Great American Merchandiser

We grumbled loudly about the merchandising tactics used in our area to promote subscriptions to *PC* magazine: a come-on mailed in an envelope that emulated an official government mailing. Dave Sullivan of Palo Alto, CA, writes to say that he got a second one of those foolers. "The latest one was from *PC Tech Journal.* It appears that Ziff-Davis is the one exhibiting all the brass.

"But I have another case of gall and guts for your scrapbook," he continues. "Ziff-Davis publishes any number of magazines that touch on computers. One of the cheaper ones is called *Computers and Electronics*, and sells for $1.75 at the newsstand. The January 1984 issue has what appears to be the first of a new series of articles on assembly language programming on the PC. It looks promising, and I'll take all the help I can get to learn assembly language.

"But the last paragraph of the article reads as follows: 'If you want to pursue learning assembly language for the IBM PC, additional articles that build on this one can be read in upcoming issues of our sister publication, *PC* magazine.'

"It takes one's breath away."

What's So Hard About CP/M Plus?

We're very fond of CP/M Plus now that we have it working well. So fond that we are trying it out on other computers besides our workhorse S-100 system. Or we would be, if anybody could get it working.

There's a Tandy Model 4P, an adorable little machine, that is patiently gathering dust while it awaits the long-delayed arrival of its CP/M Plus support from Fort Worth. Tandy promised CP/M Plus when they announced the Model 4, many months back, and they still haven't delivered. They keep promising it Real Soon Now, and they keep not shipping.

Then consider our brother-in-law Bill, whom we had all primed to buy a Morrow MD-11 as his first computer, largely because it was supposed to come with CP/M Plus. Only it won't, it seems. At least the first units of the MD-11 will be shipped with plain old CP/M 2. The dealer claims "it's real hard to make CP/M Plus work with the hard disk." To which we are inclined to say, piffle, but then what do we know?

What goes on here? Bob Blum got CP/M Plus working; we got it working. David McLanahan has had it for months on his Micro Mate (a system that, he says, deserves much wider exposure than it gets). The pubs are clear and complete; the generic parts of the system are reliable; the payoff in extra function is large. So what's the holdup?

CP/M Plus, Apple, and ALS

Advanced Logic Systems is a company that managed to make CP/M Plus work quite early. We have been trying out an Apple IIe equipped with the ALS CP/M card and CP/M Plus, and it works. It isn't as nice as CP/M Plus on a *real* computer (that should bring some mail), but to a large extent that's the fault of the Apple disk drives, not the fault of ALS or the software. There just isn't a whole lot you can do with only 170K per drive.

ALS did skimp on their implementation in one area. They completely omitted implementing the CP/M Plus character I/O table and its character I/O redirection features. We can sympathize, because character I/O in the Apple is a zoo; there are so many possible permutations of serial cards, parallel cards, ROM versions, and slot numbers that it's hard to blame them for throwing up their hands. However, they didn't help the situation when they omitted documenting character I/O in any way whatsoever. The ALS bios contains some kind of support for the AUX device, for instance, but the ALS manuals contain not even a hint as to what AUX connects to.

Since our plans for the IIe involve getting MODEM to run on it so we can exchange files between it and the big system, we had to learn something about Apple I/O under the ALS bios. The ALS hot-line seems to be overloaded; they returned one of our calls out of six. The one contact we managed to make with them didn't produce any useful technical information, but it did give us the pointer we needed.

The ALS support person pointed us to a new CP/M Plus User Group being formed around the ALS card and the Apple. The group is CP/PLUG; its president is Jim Scardelis; and you can reach the group at P. O. Box 295, Little Falls, NJ 07424. Dues are $25 a year (we think, our notes are a mess) and the first quarterly newsletter was in preparation in January.

Scardelis gave us some valuable technical hints, but told us that if we wanted more we should dial into a certain RCP/M system run by one of the club members. We did and found a gold mine of Apple/ALS material. The system is the Central New York Technical RCP/M system, Mark Howard SYSOP. It is up from 1700 to 0800 hours EST, on (315) 437-4890, and is an excellent source of technical data. Thanks to Scardelis and Howard, we got our IIe running MODEM712 in fine shape.

How It's Done

Oh, you wanted to know how you do serial I/O on the Apple? Thought you'd never ask. When you are running under the ALS CP/M card, bank 1 (the bank containing the transient program area) consists of RAM chips on the ALS card itself. Global RAM starts from E000h and is also physically on the ALS card. The original Apple address space is bank 0.

That Apple address space, as seen by a Z80 program, is remapped. Where in a real Apple the various I/O cards are memory mapped beginning at C000h, the Z80 sees them at 6000h in bank 0. So what you have to do is to find out which memory-mapped addresses your Apple serial card is supposed to appear at (somewhere in C0*xx*h). Under CP/M Plus, it will appear at 60*xx*h in bank 0.

Now all you have to do is find a way to read and write those bytes of bank 0 from a program running in bank 1. Mark Howard has written a Resident System Extension (RSX) that will do that for you. If it has been installed, you will be able to call it with a bank-0 address and get back the value of, say, the status port of your serial card. You write a MODEM7 overlay to make these calls and, voila, you are on the air. Telescoped into that "voila," of course, are about thirteen hours of sweat and hair-pulling.

Our April Item

We are in receipt of a letter from one T. Heintern of Milpitas, CA. Despite his slightly stilted English (Heintern is an immigrant from Islandia), we think his story will intrigue you.

"Your readers might be interested in the activities of two of my co-workers, or really ex-co-workers. I am, and they were, employed by a well-known maker of magnetic media here in Silicon Valley.

"At any rate, my ex-colleagues recently left our beneficent company and struck out on their own. They were remarkably close-mouthed about what they meant to do, but everyone in our department was aware that they had been spending long lunch hours in the bar at the Red Lion Inn in San Jose, in the company of various folk who wore the kind of urban-western gear that replaces a venture capitalist's ivy-league suit when he aims to blend into the crowd 'out west.' I presume my peers found the money they needed, or at any rate the strong hint of it, because they tendered their resignations just before Comdex.

"I next saw them in Las Vegas, where they were holding some fairly lush parties in a private suite. It was also at Comdex that I began to pick up rumors of a new, even-smaller, floppy disk format that (everyone told me) was about to burst on the scene. 'Forget the three-inchers,' I heard one fellow tell another in the midst of the crowd at the Microsoft Windows demo. 'It's almost standardized, so it's dead, right? The big thing is gonna be the CPS . . . .' Here the crowd pressed me away from them. Had they said CPS? I wasn't sure. Later I heard similar references to a 'Colt.'

"Back home I was swapping rumors with the other folk who'd attended the convention, and they too had heard rumors of a new diskette – and that it was to be of domestic, not Japanese, manufacture. Best of all, one had picked up an indication of its size: it was to be of 3.8 cm diameter.

"Well, I was mulling all this over when it all fell into place in my mind: CPS, Colt, 3.8 cm – a Colt Police Special is a 38-calibre pistol! Just as 'Winchester' was the code name for the IBM 3030, 'Colt' was a code name for a 3.8 cm diskette. Three point eight centimeters comes to about one and a half inches, I thought. And the light dawned!

"There flashed before my mind's eye the picture of our production floor: The great sheets of mylar spinning off the drums and under the punches . . . the thousands of 3.8 cm mylar circles fluttering down in drifts from the 8-inch floppies-to-be . . . our janitors sweeping them up . . . our teams of diligent housewives bundling them into tidy stacks to be piled in building-high columns in the insulation space of our great Eastern wall.

"I dashed outside and rapped on the wall – hollow! Our insulation, years of 3.8 cm floppy-punchings, was gone!

"I have, of course, notified corporate management, who are taking legal advice. However, I wish to warn your readers. Shun the new diskette format! After all, you've already paid for the doughnuts. Will you now buy the holes?" DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 190.**

"Q-PRO 4 blows dBASE II away

We now complete complex applications in weeks instead of months."

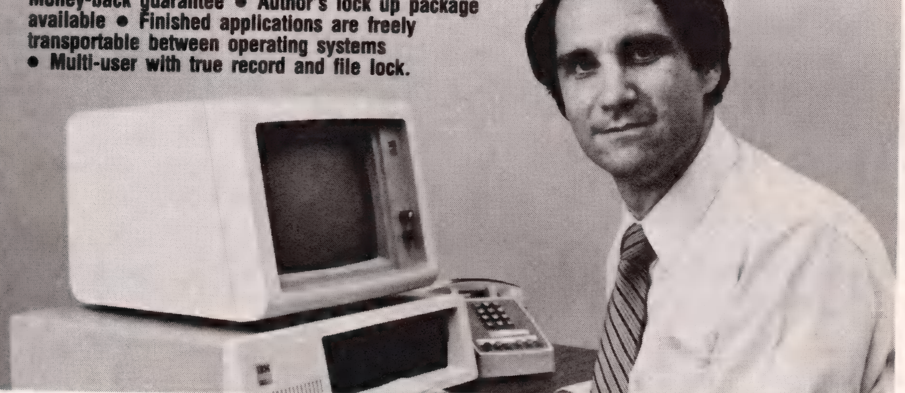
says Q-PRO 4 user, Richard Pedrelli, President, Quantum Systems, Atlanta, GA

"As a dBASE II beta test site the past two years, we were reluctant to even try Q-PRO 4. Now we write all our commercial applications in Q-PRO 4. We find it to be an order of magnitude more powerful than dBASE II.

We used Q-PRO 4's super efficient syntax to complete our Dental Management and Chiropractic Management Systems much faster. Superb error trap and help screen capabilities make our finished software products far more user friendly, too.

In my estimation, any application programmer still using outdated 3rd generation data base managers or worse, a 2nd generation language like BASIC, is ripping himself off."

Runs with PCDOS, MS-DOS, CP/M, MP/M, CP/M86, MP/M86, TurboDOS, MmmOST, MUSE, and NSTAR.
PRICE: 8-bit single-user – $395; 8-bit multi-user and 16-bit single-user – $595; 16-bit multi-user – $795.
Money-back guarantee • Author's lock up package available • Finished applications are freely transportable between operating systems • Multi-user with true record and file lock.



For Q-PRO 4 demonstration, go to nearest MicroAge store or other fine dealer.

quic·n·easi products inc.

136 Granite Hill Court, Langhorne, PA 19047 **(215) 968-5966** Telex 291-765

CP/M, MP/M, CP/M86, and MP/M86 are trademarks of Digital Research. TurbDOS, MmmOST, MUSE, NSTAR, MS-DOS and PCDOS are trademarks of Software 2000, TeleVideo Systems, O.S.M., Molecular, Microsoft and IBM, respectively.

Circle **no. 49** on reader service card.

CP/M EXCHANGE

by Robert Blum

Over the past few months it has been increasingly difficult to obtain DRI's permission to publish the application notes for CP/M. Apparently a shift in personnel at DRI had slowed the approval process, and I misunderstood how freely I could use the notes for CP/M Plus. Nonetheless, DRI has given us permission to continue publishing the application notes for CP/M V2.2 and CP/M Plus. This includes the CP/M Plus BDOS patch 02, published in the February issue of *DDJ*, which dealt with the enhancement of the LRU buffering logic of CP/M Plus. It has now been approved without changes.

When I was told of DRI's decision to allow us to continue publishing the application notes I breathed a deep sigh of relief. From the onset of our problems I was concerned that DRI was beginning to develop an attitude, or attitude problem depending on how you look at it, of pulling back from directly supporting their products. Time and again I have seen the customer support function inadequately treated by sales organizations and otherwise excellent products given undeserving tarnished reputations. I am glad that we can continue to depend on DRI to directly support their products. Starting next month we will begin publishing more application notes for CP/M.

Compatibility

Manufacturers dream of being able to move from one product generation to the next without any severe software incompatibility. And in most cases they are almost completely successful. This issue must have been a painful one at DRI while forming a product strategy for CP/M Plus. If their product marketing department is subject to the type of pressures that I have seen in similar industries they were undoubtedly presented with a difficult decision.

DRI was obligated to maintain compatibility with CP/M V2.2. It would have been incomprehensible for them to even consider breaking the tie with the hundreds of thousands of V2.2 systems currently in use. Any attempt to entice the software developers, at this time, to convert their packages to a new operating system or just to make a few modifications would have been out of the question. Being the incumbent in a time when cries are heard from every street corner that this or that new machine or software package is better than the others is worth more in advertising dollars than most companies can muster.

By having available an upward migration path the current user base would be able to easily purchase, or upgrade to, the latest version of the product. Even though not prevalent in the microcomputer industry today, we will soon find software companies demanding that customers maintain their software at current levels or pay the price of nonsupport. From the manufacturers' standpoint this is the least costly method of introducing a new product because one immediately has a qualified customer base to sell to. Support costs are reduced as well.

CP/M Plus's compatibility with older versions of CP/M is maintained at a system level. All the former BDOS calls were maintained, although in many cases extra data or status information is made available for the programmer's use. To expand the system many new calls were implemented, but again, their use is entirely up to the programmer. This is as far as DRI will go in saying that CP/M Plus is compatible with V2.2. In the System Guide they refer to CP/M Plus as simply being "upward-compatible."

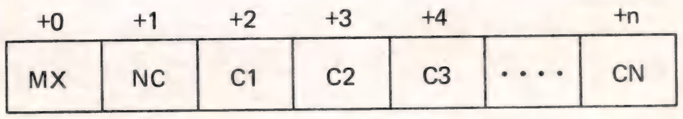
In V2.2 it was common practice to "hook" into the BIOS table to enable a program to trap status codes and other information before the BDOS had a chance to act upon it. Under CP/M Plus this practice is frowned upon and is largely unnecessary due to the new BDOS error modes and the additional status information returned from each function. Programs that do make use of the old error trapping technique will continue to run without any problems provided that system alteration does not go beyond the console and list BIOS vector points. Since these were the most-used points of alteration, DRI specified that they remain available as before.

The main reason for this restriction is the address extracted from the BIOS jump table may, and probably will not, point to a real memory address. Keep in mind that in virtual memory systems (a banked CP/M Plus system falls into this category) any memory address may or may not currently reside in the processor's physical address space. Depending on the hardware manufacturer the determination of whether a memory address is real or virtual at any moment is a combined effort of both hardware and software.

In the case of CP/M Plus, some logic sections and table structures are required to be in real (common) memory at all times due to performance and memory conflict considerations. The majority of the system, however, will be virtual. When an application program makes a call to the BDOS to request a system service, a check is made to determine whether the requested portion of the system is in real memory. If not, a portion of the application program is made virtual and the missing section of the operating system is switched to occupy the real memory address space. As complicated or confusing as this may seem, it is handled quite simply in the CP/M Plus software and should not be intimidating.

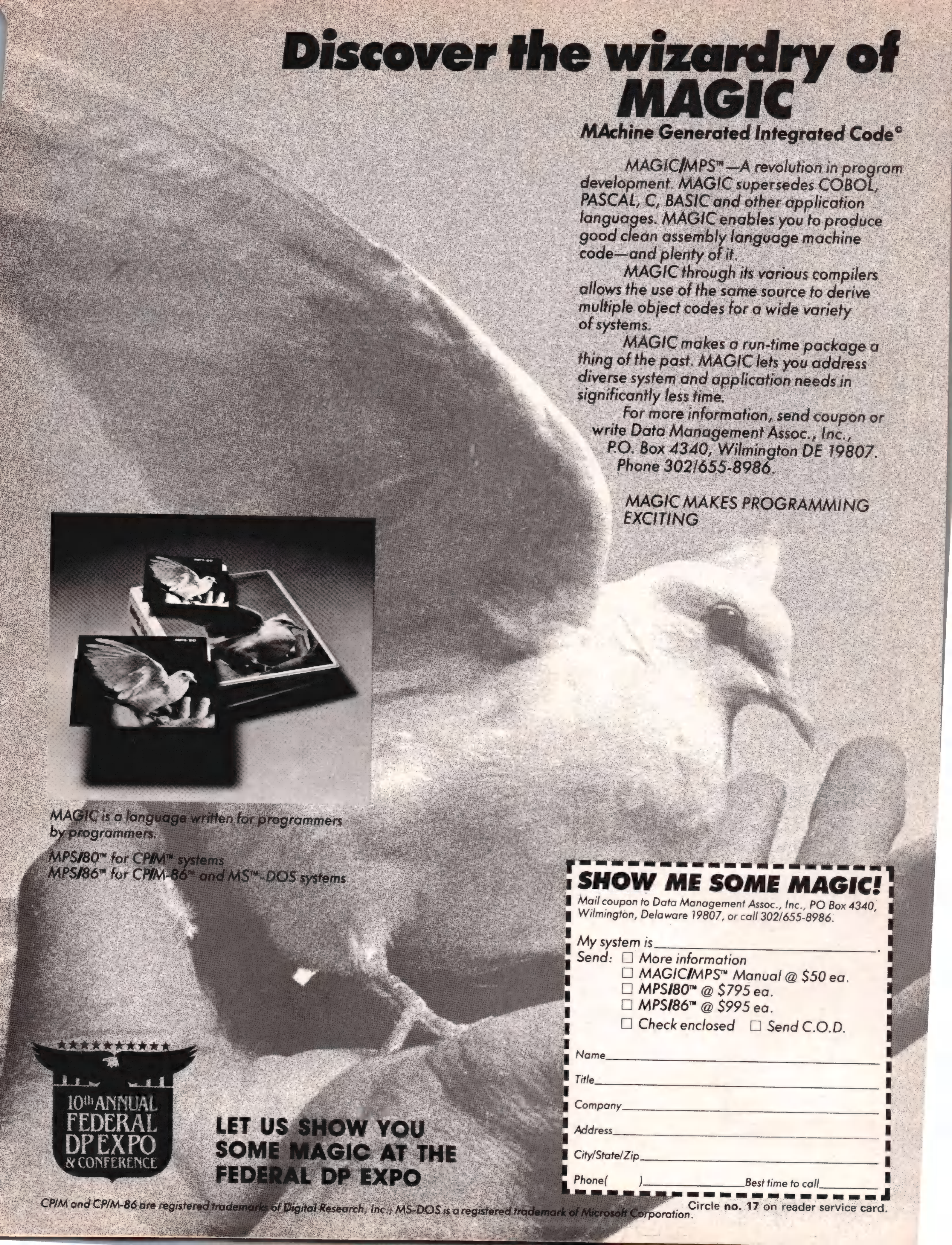
To better visualize the BIOS routines that are in virtual memory refer to Listing One (page 14). All the routines pointed to by an address constant marked by a double quote are in virtual memory. You may wish to also refer back to this column in the July and August 1983 issues. In those columns I talked more about virtual memory systems and how they are generally implemented on microcomputers and particularly under CP/M Plus.

Most of the programs I have tested under CP/M Plus ran without any problems. I have found a couple of very popular programs that refused to cooperate and caused some totally unexpected and very difficult-to-find system problems until I discontinued their use. On the other hand, some programs that I would have



Figure

Function 10 Console Input Buffer



Discover the wizardry of MAGIC

MAchine Generated Integrated Code©

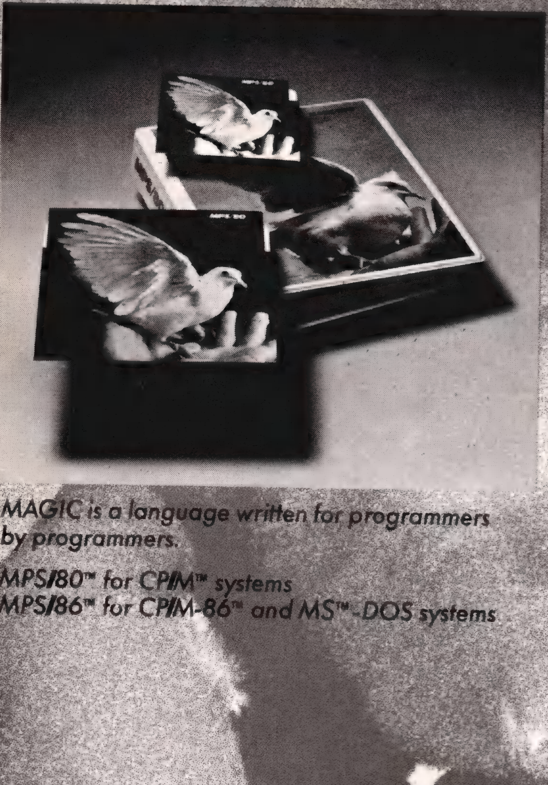
MAGIC/MPS™—A revolution in program development. MAGIC supersedes COBOL, PASCAL, C, BASIC and other application languages. MAGIC enables you to produce good clean assembly language machine code—and plenty of it.

MAGIC through its various compilers allows the use of the same source to derive multiple object codes for a wide variety of systems.

MAGIC makes a run-time package a thing of the past. MAGIC lets you address diverse system and application needs in significantly less time.

For more information, send coupon or write Data Management Assoc., Inc., P.O. Box 4340, Wilmington DE 19807. Phone 302/655-8986.

MAGIC MAKES PROGRAMMING EXCITING



MAGIC is a language written for programmers by programmers.

MPS/80™ for CP/M™ systems
MPS/86™ for CP/M-86™ and MS™-DOS systems

SHOW ME SOME MAGIC!

Mail coupon to Data Management Assoc., Inc., PO Box 4340, Wilmington, Delaware 19807, or call 302/655-8986.

My system is____________________.

Send: ☐ More information
☐ MAGIC/MPS™ Manual @ $50 ea.
☐ MPS/80™ @ $795 ea.
☐ MPS/86™ @ $995 ea.
☐ Check enclosed ☐ Send C.O.D.

Name____________________

Title____________________

Company____________________

Address____________________

City/State/Zip____________________

Phone(____)__________Best time to call__________

10th ANNUAL FEDERAL DP EXPO & CONFERENCE

LET US SHOW YOU SOME MAGIC AT THE FEDERAL DP EXPO

CP/M and CP/M-86 are registered trademarks of Digital Research, Inc.; MS-DOS is a registered trademark of Microsoft Corporation.

Circle **no. 17** on reader service card.

bet wouldn't work, do. The only suggestion that I can make to you is to ask your dealer or the manufacturer whether their programs will work under CP/M Plus. Listen to their answers very closely for key phrases such as "it should" or "I don't see why it wouldn't." If you are given an answer of this type pursue the subject further by asking whether a test was actually run and if so on what machine. I think you will find that, when backed to the wall, most of the manufacturers' representatives will admit that no direct experience is available. This is not to say that what you are looking to buy won't work; chances are it will. Just protect yourself by making sure that you can return the package if some problem does exist.

BDOS Function 10: Vastly Improved

The read console buffer function, BDOS function 10, has been vastly improved in CP/M Plus. It will now not only automatically input and edit a string from the console, but will also display the string buffer before going into edit mode.

The format of the input buffer pointed to by the DE register pair is unchanged from V2.2 (refer to Figure, page 12). The single byte, MX, is the maximum number of characters that can be placed into the buffer. And the single byte, NC, is the count of the number of characters that have been input to the buffer. The actual data that has been entered into the buffer follows the NC byte. During the data entry operation the full keyboard editing features of CP/M Plus can be used to adjust the input string.

If register pair DE is set to zero prior to calling the BDOS, function 10 assumes that an initialized input buffer is located at the current DMA address. This allows a program to put a string on the console for the user to edit. To initialize the input buffer, place the default string into the buffer following the NC byte terminated by a binary zero. When initializing the input buffer with a default string it is not necessary to set the NC byte to the number of characters in the buffer.

When a program calls the function 10 routine with an initialized buffer, function 10 operates as if the user had typed in the string. A search is made through the buffer for the binary zero terminator byte in order to count the number of active characters. This count is then placed into the NC field. The routine's attention is then tuned to allow more data to be entered from the console. At this point the user is in full keyboard editing mode and can enter more data or edit the default string. In any case hitting return will immediately send the string to the program.

As now programmed, function 10 could greatly assist in those programs that offer a default value to the user as the first choice. Refer to Listing Two (page 16) for an example of a program that makes full use of function 10's expanded capabilities.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 191.**

CP/M Exchange (Text begins on page 12)

Listing One

```
                            ; bios jump vector.

                            ; all bios routines are invoked by calling these
                            ;       entry points.

C3 0000"   ?boot:           jp     boot       ; initial entry on cold start
C3 006C'   ?wboot:          jp     wboot      ; reentry on program exit, warm start

C3 0177'   ?const:          jp     const      ; return console input status
C3 0192'   ?conin:          jp     conin      ; return console input character
C3 00DA'   ?cono:           jp     conout     ; send console output character
C3 00E6'   ?list:           jp     list       ; send list output character
C3 00E0'   ?auxo:           jp     auxout     ; send auxiliary output character
C3 0198'   ?auxi:           jp     auxin      ; return auxiliary input character

C3 006E"   ?home:           jp     home       ; set disks to logical home
C3 003F"   ?sldsk:          jp     seldsk     ; select disk drive
C3 0071"   ?sttrk:          jp     settrk     ; set disk track
C3 0077"   ?stsec:          jp     setsec     ; set disk sector
C3 007D"   ?stdma:          jp     setdma     ; set disk i/o memory address
C3 0094"   ?read:           jp     read       ; read physical block(s)
C3 00AA"   ?write:          jp     write      ; write physical block(s)

C3 0112'   ?lists:          jp     listst     ; return list device status
C3 0089"   ?sctrn:          jp     sectrn     ; translate logical to physical sector
```

(Continued on page 16)

COHERENT™ IS SUPERIOR TO UNIX* AND IT'S AVAILABLE TODAY ON THE IBM PC.

Mark Williams Company hasn't just taken a mini-computer operating system, like UNIX, and ported it to the PC. We wrote COHERENT ourselves. We were able to bring UNIX capability to the PC with the PC in mind, making it the most efficient personal computer work station available at an unbelievable price.

For the first time you get a multi-user, multitasking operating system on your IBM PC. Because COHERENT is UNIX-compatible, UNIX software will run on the PC under COHERENT.

The software system includes a C-compiler and over 100 utilities, all for $500. Similar environments cost thousands more.

COHERENT on the IBM PC requires a hard disk and 256K memory. It's available on the IBM XT, and Tecmar, Davong and Corvus hard disks.

Available now. For additional information, call or write,

Mark Williams Company
1430 West Wrightwood, Chicago, Illinois 60614
312/472-6659

Mark
Williams
Company

COHERENT is a trade mark of Mark Williams Company.
*UNIX is a trade mark of Bell Laboratories.

Circle **no. 67** on reader service card.

(Listing Continued, text begins on page 12)

Listing One

```
C3 0106'  ?conos:      jp    conost     ; return console output status
C3 017D'  ?auxis:      jp    auxist     ; return aux input status
C3 010C'  ?auxos:      jp    auxost     ; return aux output status
C3 00D2'  ?dvtbl:      jp    devtbl     ; return address of device def table
C3 0000*  ?devin:      jp    ?cinit     ; change baud rate of device

C3 00D6'  ?drtbl:      jp    getdrv     ; return address of disk drive table
C3 00CB"  ?mltio:      jp    multio     ; set multiple record count for disk i/o
C3 00CF"  ?flush:      jp    flush      ; flush bios maintained disk caching

C3 0000*  ?mov:        jp    ?move      ; block move memory to memory
C3 0000*  ?tim:        jp    ?time      ; signal time and date operation
C3 0250'  ?bnksl:      jp    bnksel     ; select bank for code execution and default dma
C3 0085"  ?stbnk:      jp    setbnk     ; select different bank for disk i/o dma operations
C3 0000*  ?xmov:       jp    ?xmove     ; set source and destination banks
```

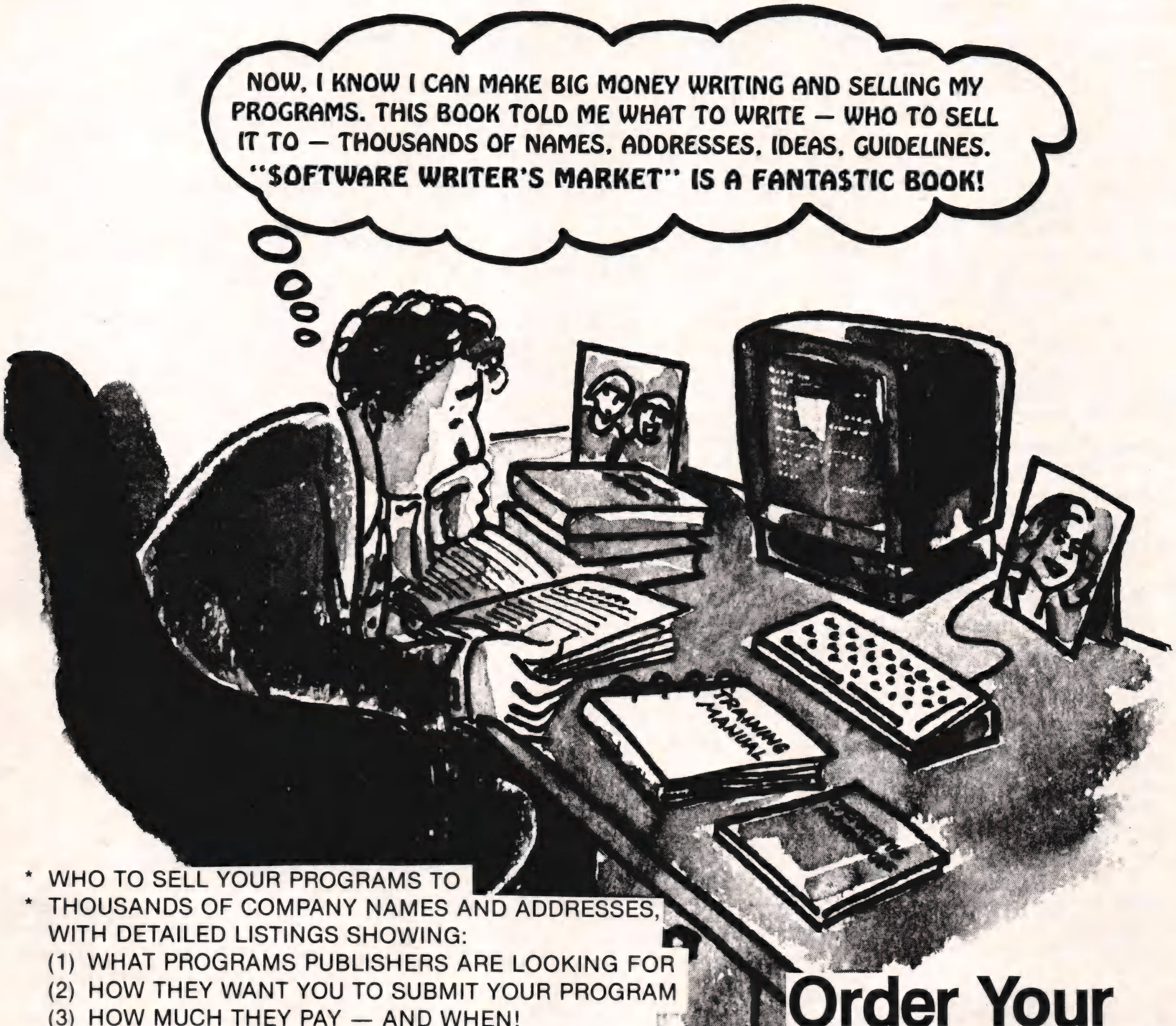
End Listing One

Listing Two

```
                .z80

bdos            equ     5               ;BDOS function call entry point

set_dma         equ     26              ;set DMA address function code
read_buffer     equ     10              ;read/edit console buffer function code

                ld      sp,stack        ;set a local stack

                ld      c,set_dma       ;set DMA address to input buffer
                ld      de,buffer       ;*
                call    bdos            ;*

                ld      c,read_buffer   ;edit string in buffer
                ld      de,0            ;*
                call    bdos            ;*

                rst     0               ;return to CP/M

buffer          equ     $
                defb    10              ;maximum number of characters allowed
                defb    5               ;number of active characters
                defb    'string',0      ;initial string - zero delimited
                defb    '    '          ;room for maximum input length

                defs    30
stack           equ     $

                end
```

End Listings

programmers READ THIS...


NOW, I KNOW I CAN MAKE BIG MONEY WRITING AND SELLING MY PROGRAMS. THIS BOOK TOLD ME WHAT TO WRITE — WHO TO SELL IT TO — THOUSANDS OF NAMES, ADDRESSES, IDEAS, GUIDELINES. "$OFTWARE WRITER'S MARKET" IS A FANTA$TIC BOOK!
TRAINING MANUAL

* WHO TO SELL YOUR PROGRAMS TO
* THOUSANDS OF COMPANY NAMES AND ADDRESSES, WITH DETAILED LISTINGS SHOWING:
  (1) WHAT PROGRAMS PUBLISHERS ARE LOOKING FOR
  (2) HOW THEY WANT YOU TO SUBMIT YOUR PROGRAM
  (3) HOW MUCH THEY PAY — AND WHEN!
* 100 CATEGORIES — FROM "ACCOUNTS RECEIVABLE" TO "GAMES" TO "VIDEO CONTROL" PROGRAMS
* HOW TO WRITE CLEAR DOCUMENTATION
* DEBUGGING TECHNIQUES

Order Your Copy Today!

Enclose check or money order for $19.95 (No C.O.D.'s) to:

IPF Publications
146 Country Club Lane
Pomona, NY 10970
(914) 354-5585

Name ....

Address ....

City............ State............ Zip............

Circle no. 57 on reader service card.

Optimizing Strings in C

by Edward McDermott

Edward McDermott, 12 Manor Haven Rd., Toronto, Ontario, Canada M6A2H9.

The C programming language offers many advantages: flexibility, speed, and portability. Routines that could take months in assembler can be written and tested much more easily in C. It supports true recursion and produces assembler language source, which allows users to optimize portions of their code.

Optimization is the process of improving the efficiency of a program; i.e., getting more bang for your buck, whether the buck is time or memory. Basically, you try to make the program smaller, faster, or both, although you sometimes have to trade size for speed or vice versa. Optimization can apply to one particular program or to a group of programs. One problem with optimization is that it takes both time and effort.

The following optimization approaches have certain advantages and disadvantages:

Improving the compiler output by improving the compiler

Improving the compiler output by tinkering

Improving the individual program's algorithm

Rewriting entire routines in assembler

To improve the compiler output, you must have the source for the compiler and must know enough to improve the code generation logic. If you have those skills, please try it and publish the results. For the rest of us mere mortals, however, this is a foolhardy exercise that can bring disaster. Besides, it takes a lot of time.

C programmers have another alternative: to recognize a commonly used pattern of code that can be improved and to write a program that tinkers with the compiler's output to improve that part of the code. This approach is very similar to the optimization portion of the Small-C compiler.

Improving an individual program by rewriting all or part of it is a perfectly valid approach, except that the payback doesn't go any farther. The rest of your programs are still the same.

The final approach is write some commonly used routines in assembler. This will speed things up and you can generalize this approach to all your programs by recompiling them. It improves both execution speed and size (to some extent). The problem is that these routines are no longer portable from machine to machine.

If you choose this last approach, its success depends on which functions you select for optimization. The ideal candidates would have the following characteristics:

Contains tight loops

Is used by almost every program

Is small enough to easily rework

C is an ideal language for editors, compilers, operating systems and so on; that was what C was designed for. Although every one of these applications involves a good deal of string handling, C has no string manipulation verbs. All string handling is done on a character-by-character basis, often by very small functions, which contain some very tight loops.

Another aspect to consider is any special commands of the CPU you are using. Can any functions built into the chip be taken advantage of by another means? The Z80 chip has a few commands that are unavailable on the 8080, specifically: LDIR, LDDR, LDI, LDD, CPIR, CPDR, CPI, and CPD. Each of these instructions is the equivalent of a number of 8080 instructions and, properly used, can be substituted for a number of instructions in C.

For example, the LDIR command takes the contents of the memory position pointed at by the HL register pair, transfers those contents to the memory position specified by the DE register pair, increments HL and DE, and decrements BC until the BC register pair is zero. Perhaps this makes it a little clearer:

```
while (--BC) *DE++ = *HL++;
```

Does that seem familiar? It is, in fact, almost identical to the STRCPY function in *The C Programming Language* by Kernighan and Ritchie. The principal difference lies in the delimitation of the loop.

If you run on a Z80 chip, string manipulation is a natural choice. But the question that haunts all optimization efforts still remains: Is the payback worth the effort? To try and answer that question we chose to optimize the following four functions:

STRCPY – copy a string from one place to another

STRINIT – initialize a string to a specified value

STRLEN – find the length of a string

CMATCH – find a character within a string

Since the Small-C compiler produces assembler code, our first approach was to review its output. In small loops the compiler produces code that does not optimally use the registers, so we completely rewrote the functions; most of the speed gain can be attributed to this rewrite. This optimization should apply to 8080-based machines as well.

The accompanying Listing (page 20) gives the new routines in Z80 code. We chose Z80 both to take advantage of its special OP codes and to emphasize the major disadvantage of this type of optimization: the loss of portability. Readers, however, can adapt these routines to the 8080 environment fairly easily, for use with C or other languages.

Usually you judge the success of an optimization on the following criteria:

Is it faster and if so how much?

Is it smaller and if so how much?

Does it suggest other methods of improvement?

Was it worth the time and effort to do?

Faster

We acquired the timings here by executing each command 10,000 times on a string of 30 characters. The first set of timings is for the driver program, without any string manipulation at all, to see the impact of the code used to handle the test. The second set is using C routines, compiled under the Small-C compiler, version 2. The third set of timings is for the same program but using the assembler routines. The timings are as follows:

Null test	3 sec.
Small C (v2)	195 sec.
Assembler	16 sec.

We also compared the CMATCH routine to the rest, since this routine is significantly longer and more involved in both languages:

Null test	1 sec.
Small C (v2)	90 sec.
Assembler	6 sec.

If you remove the constant load of the code to drive the test program, you find the new routines are 14.8 times faster. Skeptics can ignore the impact of the loops and still end up with new routines that are 12.2 times faster.

Smaller

The memory saving were illustrated by the following:

Language	**Size**
Small C(v2)	254 bytes
Assembler	116 bytes

The assembler code for the four routines was less than half the size of the comparable C code. However, saving 138 bytes is not going to make anyone jump for joy. With fine tuning, this figure could be improved, but once again it is a matter of cost versus gain.

Other Improvements

Our first thought was to optimize other small routines for handling strings. But part of the savings had come from using all three register pairs and keeping the values in the registers throughout the loop. Where the number of parameters exceeds three (one for each of BC, DE, and HL register pairs), the potential savings drop and the effort begins to rise.

Another improvement would be using these optimized routines as heavily as possible. That has some major implications for the style of C code a programmer uses. You would have to build up your functions out of smaller optimized functions instead of writing complex character-by-character processes.

Some other potential candidates for optimization are as follows:

- Convert to upper case and convert to lower case
- Find string
- Compare string
- Extract part of a string

All string manipulation functions should not necessarily be optimized. The gain in speed through improving screen and printer routines is hidden by the limited transmission speed to these devices for most small systems; you won't see any improvement. Most likely your disk I/O routines are already in assembler to interface the language to your particular operating system.

By the way, the STRCPY and STRINIT functions were designed to return the address of the resulting string. This may seem an unnecessary complication, but it does allow the programmer to nest functions within each other when writing the C code.

Time and Effort

Whether the optimization was worth the time and effort depends on how you value your time and on your requirements for speed and/or memory. My personal expectation for speed improvement was somewhere in the order of five times faster. The improvement, however, was between 12 and 15 times faster. Since creating these routines, testing them, correcting them, and correcting them again resulted in more speed than I expected, the optimization proved successful. You have to decide if it's worth your time to type these routines in and use them in your code.

The improvement you can achieve by converting small routines into assembler is significant, if not startling. Leaving the original C routine as a comment in the assembler code lets you combine the best of both worlds: the most efficient routine and the original portable routine. Any small, tight looping function that is concerned with three or fewer variables and is commonly used is an ideal candidate. However, to get the most out of such an optimization, you must go on to build more complex functions out of these smaller ones. DDJ

(Listing begins on page 20)

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 192.**

Editor's note on some enhancements: Any comparison of the accumulator to 0 can be replaced with an OR A command, and any initialization of the accumulator can be replaced with an XOR A command. This improves speed and reduces memory requirements.

Some places where one could make the code tighter are perhaps less obvious. In lines 50–54 of the listing, the following command sequence could replace the existing ones:

```
LD   A,B
OR   B
JR   Z,STRIN2
```

Likewise, lines 97–101 could be replaced with

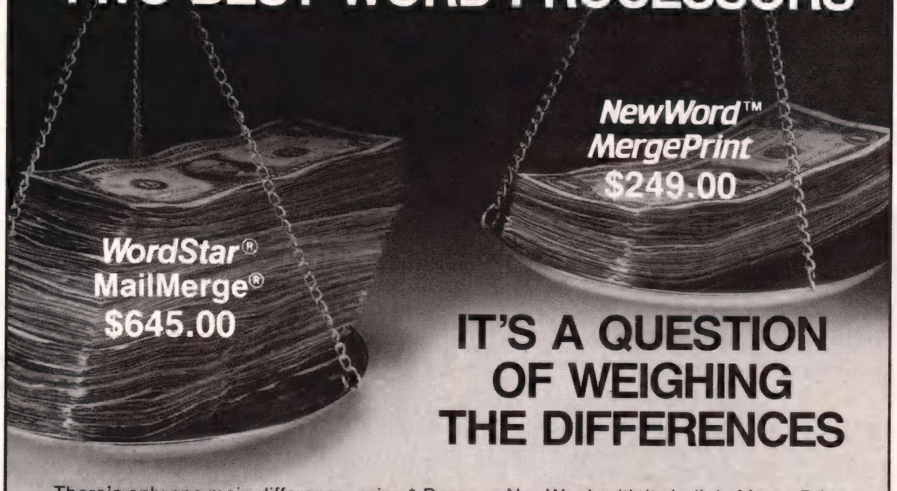
```
LD   A,B
OR   C
JR   Z,CMA1
```

and lines 103–106 could be replaced with

```
LD   A,B
OR   C
JR   NZ,CMAX1
```

In lines 104 and 106, the author used JP instructions instead of JRs. While JR is probably the better instruction, JP is safer in the case of the author's assembler and if one is uncertain about the distance to the branch location. In this case, JR should work quite well.

WHEN IT COMES TO THE
TWO BEST WORD PROCESSORS



IT'S A QUESTION
OF WEIGHING
THE DIFFERENCES

There's only one main difference: price.* Because NewWord, with its built-in Merge Print, is keystroke, command and file compatible with MicroPro's WordStar®/MailMerge®.

NewWord also offers advanced design features like Unerase deleted text, automatically changing ruler lines, multiple line Headers and Footers, and on-screen display of boldfacing and underlining. NewWord is demonstrably superior on your dot matrix printer, supporting microjustification and variable line heights/character widths.

Make a comparison. NewWord is lighter only on your checkbook.

*Manufacturer's suggested retail

Call us today, toll-free **800-832-2244**
(In California, call 800-732-2311)

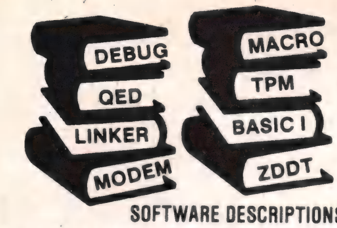
ROCKY MOUNTAIN SOFTWARE SYSTEMS
1280-C NEWELL AVE., SUITE 1052; WALNUT CREEK, CA 94596

WordStar and MailMerge are registered trademarks of MicroPro International, Inc. NewWord and Newstar are trademarks of Newstar Software, Inc.

Optimizing Strings in C (Text begins on page 18)

```
#asm
;/*
.Z80

;strcpy(s,t) char *s, *t;
;          { int ret; ret = *s; while (*s++ = *t++); return(ret)   ; }

STRCPY::
          POP      BC                   ; pop return address
          POP      HL                   ; pop address of t
          POP      DE                   ; pop address of s
          PUSH     DE                   ; restore stack
          PUSH     HL
          PUSH     BC
          PUSH     DE                   ; save address of start of s

STRLP:    LD       A,(HL)               ; transfer loop
          CP       00                   ; test for zero in t
          JR       Z,STREXT             ; if so exit
          LDI                           ; move incrementing
                                        ; s++ = t++
          JR       STRLP                ; continue loop

STREXT:   LD       A,00                 ; zero final bite in s
          LD       (DE),A
          POP      HL                   ; return original address of s
          RET

;strinit(s,c,i ) char *s; char c ; int i ;
;          { int ret; ret = *s;
;          while (i-- ) *s++=c;
;          return(ret) ; }

STRINIT::
          POP      IX                   ; pop return address
          POP      BC                   ; pop i  (len for init)
          POP      DE                   ; pop c  ( init character
          POP      HL                   ; POP ADDRESS OF STRING
          PUSH     HL                   ; restore stack
          PUSH     DE
          PUSH     BC
          PUSH     IX
          PUSH     HL                   ; save address of start of s
          LD       A,E                  ; LOAD A  with c character
          LD       (HL),A               ; store character in begin of s
          POP      DE                   ; SET DE to begin of s
          PUSH     DE
          INC      DE                   ; point DE to next bye
          DEC      BC                   ; REDUCE FOR CHAR TAKEN
          LD       A,00
          CP       C               ; MAKE SURE BC > 00
```

(Continued on page 22)



Z80 Software

SOFTWARE DESCRIPTIONS

TPM (TPM I) - $80 A Z80 only operating system which is capable of running CP/M programs. Includes many features not found in CP/M such as independent disk directory partitioning for up to 255 user partitions, space, time and version commands, date and time, create FCB, chain program, direct disk I/0, abbreviated commands and more! Available for North Star (either single or double density), TRS—80 Model I (offset 4200H) or II, Versafloppy I, or Tarbell I.

TPM-II - $125 An expanded version of TPM which is fully CP/M 2.2 compatible but still retains the extra features our customers have come to depend on. This version is super FAST. Extended density capability allows over 600K per side on an 8" disk. Available preconfigured for Versafloppy II (8" or 5"), Epson QX-10, Osborne II or TRS-80 Model II.

CONFIGURATOR I

This package provides all the necessary programs for customizing TPM for a floppy controller which we do not support. We suggest ordering this on single density (8SD).
Includes: TPM-II ($125), Sample PIOS (BIOS) SOURCE ($FREE), MACRO II ($100), LINKER ($80), DEBUG I ($80), QED ($150), ZEDIT ($50), TOP I ($80), BASIC I ($50) and BASIC II ($100)
$815 Value **NOW $250**

CONFIGURATOR II

Includes: TPM-II ($125), Sample PIOS (BIOS) SOURCE ($FREE), MACRO II ($100), MACRO III ($150), LINKER ($80), DEBUG I ($80), DEBUG II ($100), QSAL ($200), QED ($150), ZTEL ($80), TOP II ($100), BUSINESS BASIC ($200) and MODEM SOURCE ($40) and DISASSEMBLER ($80)
$1485 Value **NOW $400**

MODEL I PROGRAMMER

This package is only for the TRS-80 Model I. Note: These are the ONLY CDL programs available for the Model I. It includes: TPM I ($80), BUSINESS BASIC ($200), MACRO I ($80), DEBUG I ($80), ZDDT ($40), ZTEL ($80), TOP I ($80) and MODEM ($40)
$680 Value **NOW $175**

MODEL II PROGRAMMER

This package is only for the TRS-80 Model II. It includes: TPM-II ($125), BUSINESS BASIC ($200), MACRO II ($100), MACRO III ($150), LINKER ($80), DEBUG I ($80), DEBUG II ($100), QED ($150), ZTEL ($80), TOP II ($100), ZDDT ($40), ZAPPLE SOURCE ($80), MODEM ($40), MODEM SOURCE ($40) and DISASSEMBLER ($80)
$1445 Value **NOW $375**

BASIC I - $50, a 12K+ basic interpreter with 7 digit precision.

BASIC II - $100, A 12 digit precision version of Basic I.

BUSINESS BASIC - $200, A full disk extended basic with random or sequential disk file handling and 12 digit precision (even for TRIG functions). Also includes PRIVACY command to protect source code, fixed and variable record lengths, simultaneous access to multiple disk files, global editing, and more!

ACCOUNTING PACKAGE - $300, Written in Business Basic. Includes General Ledger, Accounts Receivable/Payable, and Payroll. Set up for Hazeltine 1500 terminal. Minor modifications needed for other terminals. Provided in unprotected source form.

MACRO I - $80, A Z80/8080 assembler which uses CDL/TDL mnemonics. Handles MACROs and generates relocateable code. Includes 14 conditionals, 16 listing controls, 54 pseudo-ops, 11 arithmetic/logical ops, local and global symbols, linkable module generation, and more!

MACRO II - $100, An improved version of Macro I with expanded linking capabilities and more listing options. Also internal code has been greatly improved for faster more reliable operation.

MACRO III - $150, An enchanced version of Macro II. Internal buffers have been increased to achieve a significant improvement in speed of assembly. Additional features include line numbers, cross reference, compressed PRN files, form feeds, page parity, additional pseudo-ops, internal setting of time and date, and expanded assembly-time data entry.

DEVELOPER I

Includes: MACRO I ($80), DEBUG I ($80), ZEDIT ($50), TOP I ($80), BASIC I ($50) and BASIC II ($100)
$440 Value **NOW $150**

DEVELOPER II

Includes: MACRO II ($100), MACRO III ($150), LINKER ($80), DEBUG I ($80), DEBUG II ($100), BUSINESS BASIC ($200), QED ($150), TOP II ($100), ZDDT ($40), ZAPPLE SOURCE ($80), MODEM SOURCE ($40), ZTEL ($80), and DISASSEMBLER ($80).
$1280 Value **NOW $350**

DEVELOPER III

Includes: QSAL ($200), QED ($150), BUSINESS BASIC ($200), ZTEL ($80) and TOP II ($100)
$730 Value **NOW $300**

COMBO

Includes: DEVELOPER II ($1280), ACCOUNTING PACKAGE ($300), QSAL ($200) and 6502X ($150)
$1930 Value **NOW $500**

LINKER - $80, A linking loader for handling the linkable modules created by the above assemblers.

DEBUG I - $80, A tool for debugging Z80 or 8080 code. Disassembles to CDL/TDL mnemonics compatible with above assemblers. Traces code even through ROM. Commands include Calculate, Display, Examine, Fill, Goto, List, Mode, Open File, Put, Set Wait, Trace, and Search.

DEBUG II - $100, A superset of Debug I. Adds Instruction Interpreter, Radix change, Set Trap/Conditional display, Trace options, and Zap FCB.

6502X - $150, A 6502 cross assembler. Runs on the Z80 but assembles 6502 instructions into 6502 object code! Similar features as our Macro assemblers.

QSAL - $200, A SUPER FAST Z80 assembler. Up to 10 times faster than conventional assemblers. Directly generates code into memory in one pass but also to offset for execution in its own memory space. Pascal like structures: repeat...until, if...then...else, while...do, begin...end, case...of. Multiple statements per line, special register handling expressions, long symbol names, auto and modular assembly, and more! This one uses ZILOG Mnemonics.

QED - $150, A screen editor which is both FAST and easy to learn. Commands include block delete, copy, and move to a named file or within text, repeat previous command, change, locate, find at start of line, and numerous cursor and window movement functions. Works with any CRT having clear screen, addressable cursor, clear to end of line, clear to end of screen, and 80X24.

ZTEL - $80, An extensive text editing language and editor modelled after DEC's TECO.

ZEDIT - $50, A mini text editor. Character/line oriented. Works well with hardcopy terminals and is easy to use. Includes macro command capability.

TOP I - $80, A Text Output Processor for formatting manuals, documents, etc. Interprets commands which are entered into the text by an editor. Commands include justify, page number, heading, subheading, centering, and more.

TOP II - $100, A superset of TOP I. Adds: embedded control characters in the file, page at a time printing, selected portion printing, include/merge files, form feed/CRLF option for paging, instant start up, and final page ejection.

ZDDT - $40, This is the disk version of our famous Zapple monitor. It will also load hex and relocatable files.

ZAPPLE SOURCE - $80, This is the source to the SMB ROM version of our famous Zapple monitor. It can be used to create your own custom version or as an example of the features of our assemblers. Must be assembled using one of our assemblers.

MODEM - A communication program for file transfer between systems or using a system as a terminal. Based on the user group version but modified to work with our SMB board or TRS-80 Models I or II. You must specify which version you want.

MODEM SOURCE - $40, For making your own custom version. Requires one of our Macro Assemblers.

DISASSEMBLER - $80, Does bulk disassembly of object files creating source files which can be assembled by one of our assemblers.

HARDWARE

S-100 — SMB II Bare Board $50, "System Monitor Board" for S-100 systems, 2 serial ports, 2 parallel ports, cassette interface, 4K memory (ROM, 2708 EPROM, 2114 RAM), and power on jump. When used with Zapple ROM below, it makes putting a S-100 system together a snap.
Zapple ROM $35, Properly initializes SMB I/II hardware, provides a powerful debug monitor.

IBM PC — Big Blue Z80 board $595, Add Z80 capability to your IBM Personal Computer. Runs CP/M programs but does not require CP/M or TPM. Complete with Z80 CPU, 64K add on memory, serial port, parallel port, time and date clock with battery backup, hard disk interface, and software to attach to PC DOS and transfer programs. Mfr'd by QCS.
50% Discount on all CDL software ordered at the same time as a Big Blue (and for the Big Blue).

APPLE II — Chairman Z80 $345, Add Z80 capability to your Apple II/II Plus computer. Runs CP/M programs with our more powerful TPM. Includes 64K memory add on (unlike the competition this is also useable by the 6502/DOS as well as the Z80), TPM, QSAL assembler, QED Screen Editor, and Business Basic. Mfr'd by AMT Research.
Apple Special $175, Buy the Apple Z80 Developer at the same time as the "Chairman" and pay only $175 instead of $325.

APPLE Z80 DEVELOPER

Includes: 6502X ($150), MACRO II ($100), MACRO III ($150), QSAL ($200), QED ($150), LINKER ($80), DEBUG I ($80), DEBUG II ($100), ZDDT ($40) and BUSINESS BASIC ($200)
VALUE: $1250 **NOW $325**
$175 when purchased with AMT "Chairman" Board

DISK FORMATS

When ordering software specify which disk format you would like.

CODE	DESCRIPTION
8SD	8" IBM 3740 Single Density (128 bytes/26 sectors/77 tracks)
8DD	8" Double Density (256 bytes/26 sectors/77 tracks)
8XD	8" CDL Extended Density (1024 bytes/8 sectcr/77 traceks = 616K)
5SD	5.25" Single Density (TRS80 Model I, Versafloppy I, Tarbell I)
5EP	5.25" Epson Double Density
5PC	5.25" IBM PC Double Density
5XE	5.25" Xerox 820 Single Density
5OS	5.25" Osborne Single Density
5ZA	5.25" Z80 Apple (Softcard compatible)

TPM INFO When ordering TPM I or II, in addition to Disk Format, please specify one of the following codes:

	CODE	DESCRIPTION
TPM I:	NSSD/H	North Star Single Density for Horizon I/O
	NSSD/Z	North Star Single Density for Zapple I/O
	NSDD/H	North Star Double Density for Horizon I/O
	NSDD/Z	North Star Double Density for Zapple I/O
	TRS80-I	TRS-80 Model I (4200H Offset)
	TRS80II	TRS-80 Model II
	VI8	Versafloppy I 8"
	VI5	Versafloppy I 5.25"
TPM-II:	VII8	Versafloppy II 8" (XD)
	VII5	Versafloppy II 5.25"
	TRS80II	TRS-80 Model II (XD)

Prices and Specifications subject to change without notice.
TPM, Z80, CP/M, TRS80 are trademarks of CDL, Zilog, DRI and Tandy respectively.

ORDERING INFORMATION:

VISA/MasterCard/C.O.D.
Call or Write With Ordering Information....

MasterCard
VISA

OEMS:

Many CDL products are available for licensing to OEM's. Write to Carl Galletti with your requirements.

Dealer Inquiries Invited.

For Phone Orders ONLY Call Toll Free...
1-(800) 458-3491 (Except Pa.)
Ask For Extension #15

For information and Tech Queries call
(609) 599-2146

Computer Design Labs
342 Columbus Avenue/Trenton, NJ 08629

Circle **no. 8** on reader service card.

Optimizing Strings in C (Listing Continued, text begins on page 18)

```
        JR      NZ,STRIN1
        CP      B
        JR      Z,STRIN2

STRIN1: LDIR                            ; propogate it through string
STRIN2: POP     HL                      ; get address of s to return
        RET
;/*  strlen -- return length of string s (page 98) */
;strlen(s) char *s;
;       { int p ;  p =s;
;         while (*s) ++s  ;
;         return (s-p);}

STRLEN::
        POP     BC                      ; pop return address
        POP     HL                      ; pop address of s
        PUSH    HL                      ; restore stack
        PUSH    BC
        PUSH    HL                      ; save address of start of s
        LD      BC,0FFFFH               ; BC IS BYTE COUNT (decremented)
        LD      A,00                    ; searching for a  00
        CPIR                            ; equivalent of while (HL++)
        POP     DE                      ; DE = s
        SBC     HL,DE                   ; HL = HL - DE
        DEC     HL                      ; correction for count last char
        RET                             ; RETURN (HL);

;cmatch(s,p,i )                             /* find first p  in str s */
;         char s[] ; int i, p ;
;         { if (i > strlen(s)) return(0);
;         while (s[i] != 0 )
;               { if (p == s[i++]) return  (i); }
;         return 0; }

CMATCH::
        POP     IX                      ; pop return address
        POP     BC                      ; pop i  (indent for init)
        POP     DE                      ; pop p  (search character
        POP     HL                      ; POP ADDRESS OF STRING s
        PUSH    HL                      ; restore stack
        PUSH    DE
        PUSH    BC
        PUSH    IX

        PUSH    BC                      ; save BC for count of bytes
        LD      A,0                     ; LOAD A  with 0
        CP      B                       ; IF BC == 0  THEN GO TO CMA1
        JR      NZ,CMA0                 ;
        CP      C
        JR      Z,CMA1
CMA0:   CPIR                            ; check 00  before end of BC
        CP      B                       ; IF (BC != 0 ) return 0
        JP      NZ,CMAX1
        CP      C
```

```
                JP      NZ,CMAX1
                                        ; else continue
        CMA1:   POP     BC              ; restor BC = i for offest count
                                        ; i is within  the string ?
        CMA2:   INC     BC             ; count bytes
                LD      A,(HL)
                CP      00              ; end of string?
                JR      Z,CMAX
                CP      E               ; check search character
                JR      Z,CMAE         ; if (HL = p) contin
                INC     HL             ; HL ++
                JR      CMA2

        CMAE:   LD      H,B            ; return (HL = BC)
                LD      L,C
                RET

                ; i is beyond the string end
        CMAX1:  POP     BC               ; restore stack
        CMAX:   LD      HL,00H           ;RETURN (NULL);
                RET
        #endasm
```

End Listing

Six Times Faster!

Super Fast Z80 Assembly Language Development Package

Z80ASM

- Complete Zilog Mnemonic set
- Full Macro facility
- Plain English error messages
- One or two pass operation
- Over 6000 lines/minute
- Supports nested INCLUDE files
- Allows external bytes, words, and expressions (EXT1 * EXT2)
- Labels significant to 16 characters even on externals (SLR Format Only)
- Integral cross-reference
- Upper/lower case optionally significant
- Conditional assembly
- Assemble code for execution at another address (PHASE & DEPHASE)
- Generates COM, HEX, or REL files
- COM files may start at other than 100H
- REL files may be in Microsoft format or SLR format
- Separate PROG, DATA & COMMON address spaces
- Accepts symbol definitions from the console
- Flexible listing facility includes TIME and DATE in listing (CP/M Plus Only)

SLRNK

- Links any combination of SLR format and Microsoft format REL files
- One or two pass operation allows output files up to 64K
- Generates HEX or COM files
- User may specify PROG, DATA, and COMMON loading addresses
- COM may start at other than 100H
- HEX files do not fill empty address space.
- Generate inter-module cross-reference and load map
- Save symbol table to disk in REL format for use in overlay generation
- Declare entry points from console
- The FASTEST Microsoft Compatible Linker available

- Complete Package Includes: Z80ASM, SLRNK, SLRIB - Librarian and Manual for just $199.99. Manual only, $30.
- Most formats available for Z80 CP/M, CDOS, & TURBODOS
- Terms: add $3 shipping US, others $7. PA add 6% sales tax

For more information or to order, call:

1-800-833-3061

In PA, (412) 282-0864

Or write: SLR SYSTEMS
1622 North Main Street, Butler, Pennsylvania 16001

SLR Systems

Circle **no. 55** on reader service card.

Expert Systems and the Weather

Somebody famous once said: "This is gonna get me in trouble." Since I am about to discuss a program that is capable of predicting the weather (sometimes), I figure I'll get in trouble, but here goes.

I chose predicting the weather as the subject of an early effort at knowledge engineering using EXPERT-2, a tiny fifth-generation type language.[1] My friends in the business of weather prediction think this is somewhat amusing, but the results are quite interesting and possibly useful. I deliberately kept the knowledge engineering exercise simple since I intended to run this program in a 48K Apple II, using a Forth system with EXPERT-2 on top and the application program, the weather predictor.

Expert systems are computer programs that are specially designed to perform inferences: to prove things. EXPERT-2 is a program that allows individuals to write task programs much like BASIC lets users write programs. It is a high-level language written for performing logic inferences using rules written into the user's program. These rules may be used to solve important problems, like predicting the stock market . . . or the weather.

EXPERT-2 allows two methods for encoding knowledge and information into the task program – in this case, the weather predictor:

IF-THEN statements

Analytic subroutines

IF-THEN statements, or production rules, allow the user to direct the inference process by entering into the computer specific statements that say, "if something is true, then something else is true." The EXPERT-2 syntax is an extension of the IF-THEN syntax used to develop an animals game.[2,3] The extensions include allowing statements to be entered in a negative context (IFNOT, ANDNOT) and calling analytic subroutines.

Analytic subroutines were added to give the programmer access to the full power of the underlying Forth system. If EXPERT-2 had been written in, say, Pascal, then the analytic subroutine calls (IFRUN, IFNOTRUN, ANDRUN, ANDNOTRUN) would permit access to that environment. As it is, EXPERT-2 was written in Forth, primarily because that's the program environment in which I work best. Users of EXPERT-2, however, need not know or understand Forth; if analytic subroutines are not needed, no Forth coding is required.

The weather predictor presented here does use analytic subroutines. These routines are used in two ways:

To prompt the user to input data

To process data and return truth values to the rules

The IF-THEN rules call the appropriate analytic subroutines when answers about weather data are needed.

The IF-THEN rules contain knowledge about the weather. This knowledge is encoded as antecedents (e.g., IF barometric pressure is falling rapidly) and consequents (e.g., THEN the weather is turning bad). Consequents follow appropriate antecedents. Designing rules to correctly encode knowledge is called knowledge engineering.

Knowledge Engineering

Someone once asked me if EXPERT-2 ever adds to its own knowledge base or makes original statements. As it turns out, it often makes original statements – especially during system debugging. Early versions of EXPERT-2, in fact, were self-modifying (not by design!). Test runs were incredibly mystifying. Knowledge engineering, as I use the term here, assumes one has available a working inference machine: a program that permits inferences to be performed on a set of rules. EXPERT-2 is one of the early microcomputer-based inference machines available; I expect there eventually will be a slew of them.

Knowledge engineering is the process of defining a problem to solve (something for the expert to do, especially something useful), encoding into a program the expertise required to solve the problem, testing, and finally validating the expert program.

To encode knowledge, one needs an expert whose brains (expertise) are accessible for encoding. As it turns out, that's not a generally available commodity; experts may be around, but it takes charm, wit, and experience to get at their brains. Trying to determine what tools an expert brings to bear on a problem often gets you a textbook set of answers. Textbook answers are not necessarily what you need to solve a problem. Factors including years of experience, intuition, and a host of other issues must be explored before a knowledge engineer fully exhausts all the elements needed to solve a particular problem.

For the weather predictor, I chose to explore only books and magazines, partly because I didn't want to call in any chits I may have with the weather service and partly because I discovered some interesting prospects in the literature for home-brew weather prediction that could be translated to the EXPERT-2 system. A set of programs have been presented for computers that run BASIC.[4] These programs ask certain questions and issue weather predictions based on the answers. Because the algorithm used in the programs relates well to the general literature on weather prediction, I selected it as a representative "expert" set of rules.

For additions and changes to the rules derived from the magazine article I drew on the Heathkit Weather Training Manual, which has a section on weather forecasting.[5] Selected rules on cloud motion, for example, were used to enhance the rule base.

Analytic subroutines are then used to acquire weather data from the user and to process that data as the inference machine needs answers.

Weather patterns are quite sensitive to local terrain influences. Therefore, the responses the weather predictor offers may not be appropriate to, say, a canyon region where you live. It seems to work well where I live. I leave it as an exercise for the reader to validate the weather predictor program in his or her own area.

System Operation

Rules are typed into a disk file using the editor available with the underlying Forth system. EXPERT-2 is then loaded and compiled on top of Forth as a task. Finally, the user task (in this case, the weather predictor) is loaded on top of EXPERT-2. First Forth compiles the analytic subroutines and any variables or constants. EXPERT-2 then compiles the IF-THEN rules once the word RULES is encountered. Rule compiling ends with the word DONE.

Forth, in its usual fashion, tells you that compiling has ended by printing OK on the screen. At this time it should be

by Jack Park

Jack Park, Helion Inc., Box 445, Brownsville, CA 95919.

safe to type RUN and start the forecasting program. RUN initializes the data base by asking appropriate questions then calls EXPERT-2 to solve the problems. The problem is to prove one of four possible hypotheses, identified in the rule base by THENHYP. The four hypotheses are:

Weather is OK
Weather is improving
Weather is turning bad
Insufficient data to perform a forecast

EXPERT-2 takes the first hypothesis and begins the task of proving it. The routine that performs this task is DIAGNOSE, which is called by RUN. DIAGNOSE selects the hypothesis and passes it deeper into the EXPERT-2 program. EXPERT-2 collects all the rules that might support a proof.

If the hypothesis is "weather is OK," then all rules that have as one of their consequent fields "weather is OK" are tagged and drawn into the inference process. One such rule is:

```
IFRUN BP>30.2
ANDRUN BP-SLOW-FALL
ANDRUN WDIR
BECAUSE WINDS FROM SW, W,
    OR NW
THENHYP WEATHER OK
ANDTHEN FAIR AND WARM
    NEXT 48 HRS
```

One of the consequent fields (in this case, the first consequent) is the same as our hypothesis; in fact, this rule happens to be the one selected to define an hypothesis with THENHYP. EXPERT-2 finds this rule and tries to prove it.

The inference machine takes each of the antecedents, one at a time, and tests them. If all pass with a truth value of TRUE, the consequents are taken as true, and the proof is complete. If any antecedent fails with a truth value of FALSE, the rest of that rule is not tested. Anything that is "learned" along the way, however, is saved as a fact. If the system asks you which way the clouds are moving, for example, your answer becomes a fact so that EXPERT-2 does not have to ask you that question again.

If EXPERT-2 proves a rule, DIAGNOSE tells you about it, and the program ends with I CONCLUDE . . . . If, on the other hand, a given hypothesis cannot be proven, then the next available hypothesis is selected, and a new proof is started. The new proof has available to it everything that the system learned during all previous proof attempts (all attempts, that is, since RUN was typed). If no proof is available on any hypothesis, EXPERT-2 typically states, CANNOT PROVE ANYTHING.

The Weather Predictor

You may wonder how such a simple-looking program can do any useful weather prediction; most people ask about this. As it turns out, the program concerns itself only with large-scale frontal movements. These fronts are best visualized (but not correctly described) as large bubbles of air hundreds of miles across, sliding around over the ground, bumping into each other, and generally moving from the west toward the east. Some of these bubbles are huge low-pressure masses of air, and some are large high-pressure masses of air.

When a low-pressure mass of air approaches your area, the barometer will spot the frontal motion and indicate that the barometric pressure is falling. As that air mass leaves your area, the barometer will begin to rise. Thus, by using readings of barometric pressure and asking questions about which way the winds are blowing, the weather predictor attempts to determine which of its "known" frontal-air-mass patterns best describes the conditions present in the atmosphere. If it finds a pattern that fits, it issues a prediction. The patterns it looks for, however, must be calibrated for your area.

As a simple exercise in knowledge engineering, the weather predictor demonstrates nearly the full capabilities of EXPERT-2. The program uses information on barometric pressure, surface wind direction, and upper air wind direction (as evidenced in upper cloud motion).

PRICE BREAKTHROUGH

The wait-loss experts have done it again!

512Kbyte SemiDisk™ with SemiSpool™

$1095

Time was, you thought you couldn't afford a SemiDisk. Now, you can't afford to be without one.

	256K	512K	1Mbyte
SemiDisk I, S-100	$895	$1095	$1795
IBM PC		$1095	$1795
TRS-80 Mdl. II, CP/M		$1095	$1795
SemiDisk II, S-100		$1395	$2095

Battery Backup Unit $150
Version 5 Software Update $30

Time was, you had to wait for your disk drives. The SemiDisk changed all that, giving you large, extremely fast disk emulators specifically designed for your computer. Much faster than floppies or hard disks, SemiDisk squeezes the last drop of performance out of your computer.

Time was, you had to wait while your data was printing. That's changed, too. Now, the SemiSpool print buffer in our Version 5 software, for CP/M 2.2, frees your computer for other tasks while data is printing. With a capacity up to the size of the SemiDisk itself, you could implement an 8 Mbyte spooler!

Time was, disk emulators were afraid of the dark. When your computer was turned off, or a power outage occurred, your valuable data was lost. But SemiDisk changed all that. Now, the Battery Backup Unit takes the worry out of blackouts.

But one thing hasn't changed. That's our commitment to supply the fastest, highest density, easiest to use, most compatible, and most cost-effective disk emulators in the world.

SemiDisk.
It's the disk the others are still trying to copy.

SEMIDISK SYSTEMS, INC.
P.O. Box GG Beaverton, OR 97075 (503) 642-3100
Call 503-646-5510 for CBBS®/NW, a SemiDisk-equipped computer bulletin board. 300/1200 baud
SemiDisk, SemiSpool Trademarks of SemiDisk Systems. CP/M Trademark Digital Research

NO WAITING

Circle **no. 52** on reader service card.

FOR TRS-80 MODELS 1, 3 & 4
IBM PC, XT, AND COMPAQ

Train Your Computer to be an EXPERT!

Expert systems facilitate the reduction of human expertise to simple, English-style rule-sets, then use them to diagnose problems. "Knowledge engineers" are developing many applications now.

EXPERT-2, Jack Park's outstanding introduction to expert systems, has been modified by MMS for MMSFORTH V2.0 and up. We supply it with full and well-documented source code to permit addition of advanced features, a good manual and sample rule-sets: stock market analysis, a digital fault analyzer, and the Animal Game. Plus the benefits of MMSFORTH's excellent full-screen editor, super-fast compiling, compact and high-speed run-time code, many built-in utilities and wide choice of other application programs.

```
( Rule 1 - demo in EXPERT-2 )
IF YOU WANT EXPERT-2
ANDNOT YOU OWN MMSFORTH
THENHYP YOU NEED TO BUY
      MMSFORTH PLUS EXPERT-2
BECAUSE MMSFORTH IS REQUIRED
```

EXPERT-2 in MMSFORTH

Another exciting tool for our alternative software environment!

- Personal License (required):

MMSFORTH System Disk IBM PC	$249.95
MMSFORTH System Disk TRS-80 1, 3 or 4	$129.95

- Personal License (optional modules):

FORTHCOM communications module	$39.95
UTILITIES	$39.95
GAMES	$39.95
EXPERT-2 expert system	$69.95
DATAHANDLER	$59.95
DATAHANDLER-PLUS (for PC only)	$99.95
FORTHWRITE word processor	$175.00

- Corporate Site License Extensions .... from $1,000

Shipping/handling & tax extra.

Ask your dealer to show you the world of MMSFORTH, or request our free brochure.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617) 653-6136

Circle **no. 36** on reader service card.

The rules begin with WALL, a word that lets you clear this rule base from memory if you want to load another program (by typing FORGET WALL). Following WALL, all analytic subroutines are loaded, written in Forth. This version (Listing One, page 27) is written in an extended Forth-79 dialect and based on the 40-character line width of a standard Apple II. Readers who type this in on other systems are encouraged to explore the enormous advantages of larger line widths; some awkward statements often are created when one is constrained to a narrow screen.

Following the analytic subroutines, RULES starts EXPERT-2's rule compiler. This invokes the IF-THEN syntax that the listing illustrates.

This weather predictor tests for 19 different possible combinations of weather conditions, using 19 basic weather prediction rules. If the requested weather information you type in fits within one of the 19 possible data combinations, a prediction will be issued. If the data falls outside all possible tests, the system will issue the statement INSUFFICIENT DATA FOR A FORECAST.

Since EXPERT-2 would rather say CANNOT PROVE ANYTHING, the last rule in the set stops that statement and issues a custom proof-collapse statement. That way, if EXPERT-2 is unable to prove any of the three main hypotheses, it is offered the last hypothesis as a consolation prize. This allows you to end an unsuccessful session with any statement you wish.

Three sets of rules are available for checking on upper level clouds. These rules, which support some of the 19 main rules by asking special questions about such clouds, underscore one of the weaknesses of EXPERT-2: its inability to handle an "I don't know" answer. That answer, in fact, does not exist in the syntax. An astute knowledge engineer, however, is still burdened with the likely event that this particular answer will be needed. Weather predictor handles the "don't know" case by allowing you to answer N for no if either you cannot see the upper clouds (there may not be any clouds to see) or you haven't gone outside to look as instructed. If you answer no, the system assumes the cloud motion supports whatever rule it happens to be trying to prove at the time.

This derives from the second rule in each of the three rule groups. The second rule, in effect, says: After dealing with the instructions (MESSAGE1), IFNOT you can see the clouds THEN clouds indicate .... One of these exists for each of the three possible conditions. A true condition is forced if you answer no. I am quite sure I'm gonna get in trouble for this.

A typical terminal session with this weather predictor follows:

1. System starts by asking barometric pressure; you enter 30.2.
2. System next asks what the barometric pressure is doing; you enter the code for falling slowly.
3. System asks for the direction from which the wind blows; you enter the code for south.
4. The terminal response is:

```
NOW, WE CAN PREDICT THE WEATHER.
I MIGHT BE ASKING A FEW MORE QUESTIONS.
I DEDUCE
   RAIN WITHIN 24 HRS
I DEDUCE
   WEATHER TURNING BAD
I CONCLUDE
   WEATHER TURNING BAD
OK (Forth's usual prompt)
```

No further questions were asked in that session. This particular session included data that triggered (successfully) the following rule:

```
IFRUN BP>30.1
   (30.2)
ANDRUN BP-SLOW-FALL
   (pressure falling slowly)
ANDRUN SDIR
   (southerly direction winds)
BECAUSE WINDS FROM SE OR S
THEN RAIN WITHIN 24 HRS
ANDTHEN WEATHER TURNING
   BAD
```

This rule illustrates a number of the design methods that were used in this knowledge engineering exercise. First, the final consequent, "weather turning bad," helps direct the proof search. If you read the rules, especially the consequent fields, you will see that a search could be conducted on dozens of different possible consequents: "rain within 24 hrs," for example. If I had set as many hypotheses as there are different consequents in this program, the hypothesis field would have been enormous. Instead I chose three primary consequents as "search pruners" to limit the search requirement. All possible rules that have a particular hypothesis are collected at the same time for proof. The first rule to pass the test terminates all further tests. Other than that, "weather turning bad" is a pretty useless consequent.

A BECAUSE field is included in this rule to allow the programmer to tack explanations in where the code might otherwise be incomprehensible. For example, if the name chosen for an analytic subroutine that is called to check wind direction is not particularly descriptive, a BECAUSE statement is added. If the system asks a question in the form IS THIS TRUE?, you can answer Y for yes, N for no, or W for

"Why did you ask me that question?" It allows you to get an explanation, which is where the BECAUSE clause is handy.

The BECAUSE clause is also handy when passing around source code for programs. One of the advantages of the EXPERT-2 style program is that it is largely self-commenting. Where it is not self-commenting, you can add BECAUSE clauses.

This EXPERT-2 program's heavy use of analytic subroutines reduces the number of questions the program might be inclined to ask at the terminal. By contrast, the animals game mentioned earlier uses absolutely no analytic subroutines: whatever it learns in the proof process is gained from the terminal. Expert programs are much more interesting if they ask something at the terminal.

Going Beyond

EXPERT-2 is a high-level language for experimenters in early fifth-generation machines. It is made available in source code form, loadable on a variety of Forth systems. Using the source, this weather predictor program, and a lot of sweat equity, I expect some pretty strong weather predictor programs will emerge.

Weather prediction is largely a data processing and pattern recognition effort, one well suited to an expert system. By adding some simplified models related to atmospheric physics and coupling the program (through analytic subroutine functions) to an automatic weather station, you could perhaps create a new program for predicting weather hazards for farmers or sporting events.

Weather forecasters rate each other by a value they call "skill." Flipping a coin usually gets a skill of 0.5 or 50% right. It will be interesting to see what skill this program and its derivatives achieve. Since local weather prediction is sensitive to local geographic terrain effects, this program will need some fiddling to raise its skill, perhaps even to the 50% level.

Some thought might go into letting the program run on a pseudo-real-time basis, keeping track of its own skill. It also might be written to fiddle with its own inferences in an attempt to raise its skill: a self-learning system. You will have to make sure, however, that it does not learn how to fiddle with its skill algorithm.

References

[1] *EXPERT-2 - A Knowledge-Based Inference Program*. Available as a document or document with source code on disk from Mountain View Press, Inc. (P. O. Box 4656, Mt. View, CA 94040), Miller Microcomputer Services (61 Lake Shore Rd., Natick, MA 01760), or Helion, Inc., Box 445, Brownsville, CA 95919.

[2] Richard Duda and John Gaschnig. "Knowledge-Based Expert Systems Come of Age." *Byte*. September 1981.

[3] P. H. Winston and B. K. P. Horn. *LISP*. Addison-Wesley.

[4] George W. Miller. "Weather Forecaster." *COMPUTE!* August 1983. See also Tom Fox, *Unique Electronic Weather Projects*. Howard Sams & Co., 1978.

[5] *Weather Training Manual*. Benton Harbor, MI: Heath Company.

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 193.**

Weather Predictor (Text begins on page 24)

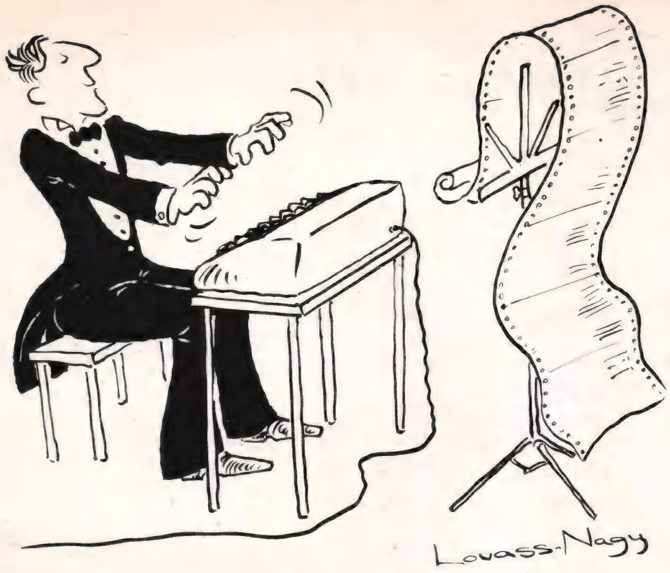
```
Screen 1
( WEATHER PREDICTOR - EXPERT-2      JP)
: WALL ; FORTH
VARIABLE MESS1             ( toggle flag )
2VARIABLE BP          ( barometric press)
VARIABLE DIR 2 ALLOT         ( dir code )
VARIABLE BP?    ( pressure trend code )

: RUN Ø MESS1 ! DIR 4 ERASE HOME CR ."
     NIMBLE WEATHER PROGRAM" CR CR   ."
THIS PROGRAM ATTEMPTS TO PROVE ONE OF
THE FOLLOWING : " CR                 ."
  THE WEATHER IS OK" CR              ."
  THE WEATHER IS IMPROVING" CR       ."
  THE WEATHER IS DETERIORATING" CR   ."
NOT ALL WEATHER CASES ARE INCLUDED IN
THE FORECAST ALGORITHM." CR CR       ."
BEFORE WE ATTEMPT A FORECAST,
I NEED SOME DATA." CR CR             ."
WHAT IS THE BAROMETRIC PRESSURE READING?
< EXAMPLE - TYPE 3Ø.2 > " INPUTD# CR CR
  BP 2!
-->
  CONTINUED ON NEXT SCREEN


Screen 2
( WEATHER PREDICTOR - EXPERT-2        JP)
                                     ."
SELECT FROM THE FOLLOWING : " CR     ."
 1 - BP IS STEADY" CR                ."
 2 - BP IS RISING SLOWLY" CR         ."
 3 - BP IS RISING RAPIDLY" CR        ."
 4 - BP IS FALLING SLOWLY" CR        ."
 5 - BP IS FALLING RAPIDLY " INPUT#
   BP? ! CR                          ."
THE WIND IS BLOWING FROM WHICH OF THE
FOLLOWING DIRECTIONS ? " CR          ."
< N, NE, E, SE, S SW, W, NW > "
  DIR 3 EXPECT CR CR                 ."
NOW WE CAN PREDICT THE WEATHER." CR ."
I MIGHT BE ASKING A FEW MORE QUESTIONS"
  DIAGNOSE ( start EXPERT-2 ) ;
-->


Screen 3
( WEATHER PREDICTOR - EXPERT-2        JP)
( analytical subroutines follow )
: BP>3Ø.2 ( -- TF )
  BP 2@ 3Ø.2 D> ;

: BP>3Ø.1 ( -- TF )
  BP 2@ 3Ø.1 D> ;

: BP<3Ø.1 ( -- TF )
  BP 2@ 3Ø.1 D< ;

: BP>29.9 ( -- TF )
  BP 2@ 2DUP 29.8 D< >R 29.8 D= R> OR ;

HEX ( now set-up direction tests )
: WDIR ( -- TF )
  DIR @ 5753 ( SW ) OVER = SWAP 57 ( W )
  OVER = SWAP 574E ( NW ) = OR OR ;
: SDIR ( -- TF )
  DIR @ 4553 ( SE ) OVER = SWAP 53 ( S )
  = OR ;
-->
```

(Continued on next page)

Weather Predictor (Listing Continued, text begins on page 24)

```
Screen 4
( WEATHER PREDICTOR - EXPERT-2        JP)
: NEDIR DIR @ 454E ( NE ) OVER = SWAP
  45 ( E ) = OR ;
: NDIR NEDIR DIR @ 4E ( N ) = OR ;
: EDIR NEDIR DIR @ 4553 ( SE ) = OR ;
: SEDIR SDIR DIR @ 45 ( E ) = OR ;
: SWDIR DIR @ 53 ( S ) OVER = SWAP
  5753 ( SW ) = OR ;
: BP-SLOW-FALL BP? @ 4 = ;
: BP-RAPID-FALL BP? @ 5 = ;
: BP-SLOW-RISE BP? @ 2 = ;
: BP-RAPID-RISE BP? @ 3 = ;
: BP-STEADY BP? @ 1 = ;
DECIMAL
-->

Screen 5
( WEATHER PREDICTOR - EXPERT-2        JP)

: MESSAGE1 MESS1 @ NOT
  IF CR                                ."
FOR THE NEXT QUESTION, YOU SHOULD STAND
OUTSIDE WITH YOUR BACK TO THE SURFACE
WIND." CR                              ."
YOU NOW MUST OBSERVE THE DIRECTION UPPER
LEVEL CLOUDS ARE MOVING." CR           ."
YOU OBSERVE THEM TO BE MOVING FROM YOUR
RIGHT, FROM YOUR LEFT, OR IN A DIRECTION
PARALLEL TO THAT WHICH YOU ARE FACING."
      CR                               ."
USE THIS INFORMATION TO ANSWER THE
FOLLOWING QUESTION. IF YOU ARE UNABLE TO
SEE THE UPPER LEVEL CLOUDS, ANSWER NO TO
THE FOLLOWING QUESTION" CR
  THEN 1 MESS1 ! TRUE ;
-->

Screen 6
( WEATHER PREDICTOR - EXPERT-2        JP)
RULES  ( start the rule compiler )
( deteriorate test using clouds )
 IFRUN MESSAGE1
 ANDIF YOU CAN SEE UPPER LEVEL CLOUDS
 ANDIF CLOUDS MOVING FROM YOUR LEFT
THEN CLOUDS INDICATE BAD WEATHER
 IFRUN MESSAGE1
 ANDNOT YOU CAN SEE UPPER LEVEL CLOUDS
THEN CLOUDS INDICATE BAD WEATHER
( improving test using clouds )
 IFRUN MESSAGE1
 ANDIF YOU CAN SEE UPPER LEVEL CLOUDS
 ANDIF CLOUDS MOVING FROM YOUR RIGHT
THEN CLOUDS INDICATE WEATHER IMPROVING
 IFRUN MESSAGE1
 ANDNOT YOU CAN SEE UPPER LEVEL CLOUDS
THEN CLOUDS INDICATE WEATHER IMPROVING
-->

Screen 7
( WEATHER PREDICTOR - EXPERT-2        JP)
( steady test using clouds )
 IFRUN MESSAGE1
 ANDIF YOU CAN SEE UPPER LEVEL CLOUDS
 AND CLOUDS MOVE PARALLEL TO DIRECTION
THEN CLOUDS INDICATE STEADY WEATHER
 IFRUN MESSAGE1
 ANDNOT YOU CAN SEE UPPER LEVEL CLOUDS
THEN CLOUDS INDICATE STEADY WEATHER
-->

Screen 8
( WEATHER PREDICTOR - EXPERT-2        JP)
 IFRUN BP>30.2
 ANDRUN BP-SLOW-FALL
 ANDRUN WDIR
 BECAUSE WINDS FROM SW, W, OR NW
THENHYP WEATHER OK
ANDTHEN FAIR AND WARM NEXT 48 HRS
 IFRUN BP>30.2
 ANDRUN BP-STEADY
 ANDRUN WDIR
 BECAUSE WINDS FROM SW, W, OR NW
THEN CONTINUED FAIR
ANDTHEN LITTLE TEMPERATURE CHANGE
ANDTHEN WEATHER OK
 IFRUN BP>30.1
 ANDRUN BP-STEADY
 ANDRUN WDIR
 BECAUSE WINDS FROM SW, W, OR NW
THEN FAIR WEATHER
ANDTHEN LITTLE CHANGE NEXT 48 HRS
ANDTHEN WEATHER OK
-->

Screen 9
( WEATHER PREDICTOR - EXPERT-2        JP)
 IFRUN BP>30.1
 ANDRUN BP-SLOW-FALL
 ANDRUN NEDIR
 BECAUSE WINDS FROM NE OR E
 AND SEASON IS SUMMER
 BECAUSE NOT WINTER
 AND CLOUDS INDICATE STEADY WEATHER
THEN RAIN MAY NOT FALL
ANDTHEN STEADY FOR SEVERAL DAYS
ANDTHEN WEATHER OK
-->

Screen 10
( WEATHER PREDICTOR - EXPERT-2        JP)
 IFRUN BP<29.8
 ANDRUN BP-RAPID-RISE
THENHYP WEATHER IMPROVING
ANDTHEN CLEARING AND COLDER
 IFRUN BP<30.1
 ANDRUN BP-SLOW-RISE
 ANDRUN SWDIR
 BECAUSE WINDS FROM S OR SW
 AND CLOUDS INDICATE WEATHER IMPROVING
THEN CLEARING WITHIN A FEW HRS
ANDTHEN FAIR NEXT SEVERAL DAYS
ANDTHEN WEATHER IMPROVING
-->

Screen 11
( WEATHER PREDICTOR - EXPERT-2        JP)
 IFRUN BP>30.1
 ANDRUN BP-SLOW-FALL
 ANDRUN EDIR
```

(Continued on page 30)



•NEW PRODUCTS•

Before Johann Sebastian Bach developed a new method of tuning, you had to change instruments practically every time you wanted to change keys. Very difficult.

Before Avocet introduced its family of cross-assemblers, developing micro-processor software was much the same. You needed a separate development system for practically every type of processor. Very difficult and very expensive.

But with Avocet's cross-assemblers, a single computer can develop software for virtually any microprocessor! Does that put us in a league with Bach? You decide.

The Well-Tempered Cross-Assembler

Development Tools That Work

Avocet cross-assemblers are fast, reliable and user-proven in over 3 years of actual use. Ask NASA, IBM, XEROX or the hundreds of other organizations that use them. Every time you see a new microprocessor-based product, there's a good chance it was developed with Avocet cross-assemblers.

Avocet cross-assemblers are easy to use. They run on any computer with CP/M* and process assembly language for the most popular microprocessor families.

5¼" disk formats available at no extra cost include Osborne, Xerox, H-P, IBM PC, Kaypro, North Star, Zenith, Televideo, Otrona, DEC.

Avocet Cross-assembler	Target Microprocessor	CP/M-80 Version	•CP/M-86 IBM PC, MSDOS** Versions•
•XASMZ80	Z-80		$250.00 each
•XASM85	8085		$250.00 each
XASM05	6805	$200.00 each	$250.00 each
XASM09	6809	$200.00 each	$250.00 each
XASM18	1802	$200.00 each	$250.00 each
XASM48	8048/8041	$200.00 each	$250.00 each
XASM51	8051	$200.00 each	$250.00 each
XASM65	6502	$200.00 each	$250.00 each
XASM68	6800/01	$200.00 each	$250.00 each
XASMZ8	Z8	$200.00 each	$250.00 each
XASMF8	F8/3870	$200.00 each	$300.00 each
XASM400	COP400	$200.00 each	$300.00 each
XASM75	NEC 7500	$500.00	$500.00
Coming soon: XASM68K...68000			

Turn Your Computer Into A Complete Development System

Of course, there's more. Avocet has the tools you need from start to finish to enter, assemble and test your software and finally cast it in EPROM:

Text Editor VEDIT -- full-screen text editor by CompuView. Makes source code entry a snap. Full-screen text editing, plus TECO-like macro facility for repetitive tasks. Pre-configured for over 40 terminals and personal computers as well as in user-configurable form.

CP/M-80 version	$150
CP/M-86 or MDOS version	$195

(when ordered with any Avocet product)

EPROM Programmer -- Model 7128 EPROM Programmer by GTek programs most EPROMS without the need for personality modules. Self-contained power supply ... accepts ASCII commands and data from any computer through RS 232 serial interface. Cross-assembler hex object files can be down-loaded directly. Commands include verify and read, as well as partial programming.

PROM types supported: 2508, 2758, 2516, 2716, 2532, 2732, 2732A, 27C32, MCM8766, 2564, 2764, 27C64, 27128, 8748, 8741, 8749, 8742, 8751, 8755, plus Seeq and Xicor EEPROMS.

(Upgrade kits will be available for new PROM types as they are introduced.)

Programmer	$429
Options include:	
• Software Driver Package --	
• enhanced features, no installation	
• required.	
• CP/M-80 Version	$ 75
• IBM PC Version	$ 95
RS 232 Cable	$ 30
8748 family socket adaptor	$ 98
8751 family socket adaptor	$174
• 8755 family socket adaptor	$135

• **G7228 Programmer by GTek** -- baud
• to 2400 ... superfast, adaptive program-
• ming algorithms ... programs 2764 in one
• minute.

• Programmer $549

• Ask us about Gang and PAL programmers.

• **HEXTRAN Universal HEX File Con-**
• **verter** -- Converts to and from Intel,
• Motorola, MOS Technology, Mostek,
• RCA, Fairchild, Tektronix, Texas
• Instruments and Binary formats.

• Converter, each version $250

Call Us

If you're thinking about development systems, call us for some straight talk. If we don't have what you need, we'll help you find out who does. If you like, we'll even talk about Bach.

CALL TOLL FREE 1-800-448-8500
(In the U.S. except Alaska and Hawaii)

VISA and Mastercard accepted. All popular disc formats now available -- please specify. Prices do not include shipping and handling -- call for exact quotes. OEM INQUIRIES INVITED.

*Trademark of Digital Research **Trademark of Microsoft

AVOCET SYSTEMS INC.™

DEPT. 4-84 DDJ
804 SOUTH STATE STREET
DOVER, DELAWARE 19901
302-734-0151 TELEX 467210

Circle no. **3** on reader service card.

MicroMotion

MasterFORTH

It's here — the next generation of MicroMotion Forth.

- Meets all provisions, extensions and experimental proposals of the FORTH-83 International Standard.
- Uses the host operating system file structure (APPLE DOS 3.3 & CP/M 2.x).
- Built-in micro-assembler with numeric local labels.
- A full screen editor is provided which includes 16 x 64 format, can push & pop more than one line, user definable controls, upper/lower case keyboard entry, A COPY utility moves screens within & between lines, line stack, redefinable control keys, and search & replace commands.
- Includes all file primitives described in Kernigan and Plauger's Software Tools.
- The input and output streams are fully redirectable.
- The editor, assembler and screen copy utilities are provided as relocatable object modules. They are brought into the dictionary on demand and may be released with a single command.
- Many key nucleus commands are vectored. Error handling, number parsing, keyboard translation and so on can be redefined as needed by user programs. They are automatically returned to their previous definitions when the program is forgotten.
- The string-handling package is the finest and most complete available.
- A listing of the nucleus is provided as part of the documentation.
- The language implementation exactly matches the one described in FORTH TOOLS, by Anderson & Tracy.
- Floating Point & HIRES options available.
- Available for APPLE II/II+/IIe & CP/M 2.x users.
- MasterFORTH – $100.00. FP & HIRES – $40.00 each (less 25% for FP & HIRES).
- Publications
  - FORTH TOOLS – $20.00
  - 83 International Standard – $15.00
  - FORTH-83 Source Listing 6502, 8080, 8086 – $20.00 each.

Contact:

MicroMotion
12077 Wilshire Blvd., Ste. 506
Los Angeles, CA 90025
(213) 821-4340

Weather Predictor

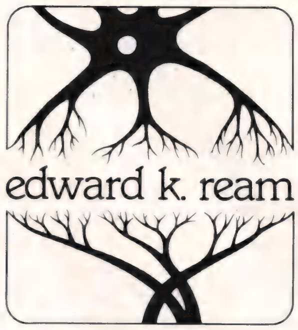
(Listing Continued, text begins on page 24)

```
 BECAUSE WINDS FROM NE, E, OR SE
 AND CLOUDS INDICATE BAD WEATHER
THEN RAIN IN 12 - 18 HRS
ANDTHEN WEATHER TURNING BAD
 IFRUN BP>30.1
 ANDRUN BP-RAPID-FALL
 ANDRUN EDIR
 BECAUSE WINDS FROM NE, E, OR SE
THEN WINDY, RAIN WITHIN 12 HRS
ANDTHEN WEATHER TURNING BAD
-->


Screen 12
( WEATHER PREDICTOR - EXPERT-2          JP)
 IFRUN BP>30.1
 ANDRUN BP-SLOW-FALL
 ANDRUN NEDIR
 BECAUSE WINDS FROM NE OR E
 AND SEASON IS WINTER
 BECAUSE NOT SUMMER
 AND CLOUDS INDICATE BAD WEATHER
THEN RAIN WITHIN 24 HRS
ANDTHEN WEATHER TURNING BAD
 IFRUN BP>30.1
 ANDRUN BP-RAPID-FALL
 ANDRUN NEDIR
 BECAUSE WINDS FROM NE OR E
 AND SEASON IS SUMMER
 BECAUSE NOT WINTER
 AND CLOUDS INDICATE BAD WEATHER
THEN RAIN WITHIN 12 - 24 HRS
ANDTHEN WEATHER TURNING BAD
-->


Screen 13
( WEATHER PREDICTOR - EXPERT-2          JP)
 IFRUN BP<29.8
 ANDRUN BP-RAPID-FALL
 ANDRUN NDIR
 BECAUSE WINDS FROM N, NE, OR E
THEN SEVERE STORM WARNING
ANDTHEN SEVERE NORTHEAST GALES
ANDTHEN WEATHER TURNING BAD
 IFRUN BP<29.8
 ANDRUN BP-RAPID-FALL
 ANDRUN SEDIR
 BECAUSE WINDS FROM E, SE, OR S
THEN SEVERE STORM WARNING
ANDTHEN RAIN OR SNOW IMMINENT
ANDTHEN WEATHER TURNING BAD
-->

Screen 14
( WEATHER PREDICTOR - EXPERT-2          JP)
 IFRUN BP<30.1
 ANDRUN BP-SLOW-FALL
 ANDRUN EDIR
 BECAUSE WINDS FROM SE, E, OR NE
 AND CLOUDS INDICATE BAD WEATHER
THEN RAIN FOR NEXT DAY OR TWO
ANDTHEN WEATHER TURNING BAD
-->
```

```
Screen 15
( WEATHER PREDICTOR - EXPERT-2        JP)
 IFRUN BP<3Ø.1
 ANDRUN BP-RAPID-FALL
 ANDRUN EDIR
 BECAUSE WINDS FROM SE, E, OR NE
 AND CLOUDS INDICATE BAD WEATHER
THEN RAIN WITH HIGH CLOUDS
ANDTHEN CLEARING WITHIN 24 HRS
ANDTHEN COOLER TEMPERATURES
ANDTHEN WEATHER TURNING BAD
-->

Screen 16
( WEATHER PREDICTOR - EXPERT-2        JP)
 IFRUN BP>3Ø.1
 ANDRUN BP-RAPID-RISE
 ANDRUN WDIR
 BECAUSE WINDS FROM SW, W, OR NW
THEN FAIR TODAY
ANDTHEN RAIN & WARMER NEXT 48 HRS
ANDTHEN WEATHER TURNING BAD
 IFRUN BP>3Ø.1
 ANDRUN BP-SLOW-FALL
 ANDRUN WDIR
 BECAUSE WINDS FROM SW, W, OR NW
THEN WARMER
ANDTHEN RAIN WITHIN 24 - 36 HRS
ANDTHEN WEATHER TURNING BAD
-->

Screen 17
( WEATHER PREDICTOR - EXPERT-2        JP)
 IFRUN BP>3Ø.1
 ANDRUN BP-RAPID-FALL
 BECAUSE RAPID FALL > .Ø6" PER HOUR
 ANDRUN WDIR
 BECAUSE WINDS FROM SW, W, OR NW
 AND CLOUDS INDICATE BAD WEATHER
THEN WARMER
ANDTHEN RAIN WITHIN 18 - 24 HRS
ANDTHEN WEATHER TURNING BAD
 IFRUN BP>3Ø.1
 ANDRUN BP-SLOW-FALL
 ANDRUN SDIR
 BECAUSE WINDS FROM SE OR S
THEN RAIN WITHIN 24 HRS
ANDTHEN WEATHER TURNING BAD
 IFRUN BP>3Ø.1
 ANDRUN BP-RAPID-FALL
 ANDRUN SDIR
 BECAUSE WINDS FROM SE OR S
THENHYP WEATHER TURNING BAD
ANDTHEN WINDY, RAIN WITHIN 12 HRS
-->

Screen 18
( WEATHER PREDICTOR - EXPERT-2        JP)
( closing statement generator )
 IFNOT WEATHER OK
 ANDNOT WEATHER TURNING BAD
 ANDNOT WEATHER IMPROVING
THENHYP INSUFFICIENT DATA FOR A FORECAST

DONE
;S
```

End Listing

RED



FULL SCREEN EDITOR
with
FULL SOURCE CODE in C
for
CP/M 68K or CP/M 80

- RED is a powerful yet simple full screen text editor for both programmers and writers.
- RED features 17 commands including block move, block copy, search and substitute.
- RED comes with *full* source code in standard C. RED works as is with the BDS C, Aztec CII and Digital Research Compilers.
- RED supports *all* features of your terminal. You tailor RED to your terminal with an easy-to-use configuration program.
- RED handles files as large as your disk.
- RED is guaranteed. If for any reason you are not satisfied with RED, your money will be refunded promptly.

Price: $95.

Call today for more valuable information:
(608) 231-2952

To order, send a check or money order to:
Edward K. Ream
1850 Summit Avenue
Madison, Wisconsin 53705

Your order will be mailed to you within one week. Sorry, I can not handle credit cards. Please do not send purchase orders unless a check is included. RED is distributed only on 8 inch CP/M format disks, but other disk formats are available through third parties.

Dealer inquiries invited.

RSA: A Public Key Cryptography System, Part II

Welcome back for the second pass. If you are like I am, you have read or scanned between twenty and forty magazines in the month that has passed since you read the first part of this article. (It keeps getting harder to keep up with technology.) Therefore, we should briefly review what we presented in Part I.

Part I Revisited

We first learned that Diffie and Hellman introduced the Public Key System (PKS) in 1976[1] and expanded on it in 1979.[2] In 1978, Rivest, Shamir, and Adleman (RSA) proposed a method of PKS implementation and introduced signing messages with digital signatures.[3]

Then we ran through a brief introduction to RATFOR (RATional FORtran), the implementation language. RATFOR, which is similar in many ways to C and Pascal, adds several structured constructs to Fortran, and the output of the RATFOR precompiler is Fortran. Sample compilation of a RATFOR "program" to source was provided in Listings One to Three.

Next we reviewed modulo arithmetic. We saw that modulo arithmetic has some advantages and disadvantages. It limits the size of the numbers that are generated, but it requires considerable extra computation. As we will see, modulo arithmetic is an integral part of the RSA cryptosystem.

Next we looked at multiple-precision arithmetic. The multiple-precision arithmetic algorithms implemented in Part I form the core of the RSA system that we will present here. Knuth's algorithms[4] were stated and Listing Four showed the RATFOR source that implemented these algorithms. Four algorithms were presented: addition, subtraction, multiplication, and division. We saw that these algorithms were very computation intensive and that optimization efforts should probably be directed at them.

by C.E. Burton

C. E. Burton, 1720 S. Deframe Court, Denver, CO 80228.

Finally, the Russian Peasant exponentiation algorithm was presented. The basic algorithm found in Knuth's book had to be expanded to include modulo operations (a requirement of RSA). The Russian Peasant algorithm is most suitable for use with large exponents, which is why it was selected. Listing Five showed the RATFOR implementation of the algorithm.

In this part we will explore the generation and testing of large prime numbers, the generation of public/private keys, the encryption/decryption process, and the use of digital signatures. However, before we get to the meat of things, we must revisit RATFOR.

RATFOR Revisited

One of the advantages of RATFOR is that it allows character and string manipulation, a shortcoming of Fortran. Most of the library functions are associated with character manipulation. Since several of these library routines are used in the programs presented in this part, we will look at them briefly.

Also, RATFOR can have a global definition file, which I have called RATDEF.RAT. RATDEF contains definitions for most of the common ASCII characters, including control characters. For example:

```
define(BIGD,68) # "D"
define(LETD,100) # "d"
define(BACKSPACE,8)
    # <BACKSPACE> character
```

These are defined because Microsoft's Fortran converts all lower-case characters to upper case, i.e., "d" translates to "D." In addition, RATDEF contains some key word definitions. For example:

```
define(character,byte)
    # make CHARACTER type equivalent to BYTE
define(EOS,-2)
    # End of String character
define(EOF,-1) # End of File character
```

The library functions that are used in the RSA-PKS routines include:

CLOSER	- closes a file opened by OPENR
EQUAL	- compares two strings for equality
ERROR	- prints a string then exits to the operating system
GETC	- waits for and gets a character from the console
GETCH	- gets a character from a Logical Unit specified in OPENR
OPENR	- opens a character type file in the specified mode and sets aside a specified buffer size
PUTCH	- sends a character to the Logical Unit specified in OPENR
PUTDEC	- converts an integer to a character string of specified length and sends the string to the console
REMARK	- sends a string to the console without a carriage return; a period terminates the string
RMRKLN	- sends a string to the console terminated by a carriage return; a period terminates the string
TYPE	- determines if a character is a DIGIT or a LETTER

So much for our revisit to RATFOR. Now, let's talk a little bit about the RSA system and what makes it so advantageous.

Public Key Systems

PKS cryptography systems are different from most other cryptography systems in that a key is made public. Other cryptography systems require that the key be kept secret; otherwise, the system is compromised. Private key methods require that both the sending and the receiving sites have copies of the key so that they can encrypt/decrypt messages. The logistics of key distribution can be very awkward and costly. For communication networks, this process is almost useless.

PKS systems do not have these problems, since everyone has access to the public (encrypting) key; however, only the receiver knows the private (decrypting) key. This dual key approach allows anyone to use the public key to encrypt a message and send it to the holder of the private key, who is the only individual that can decrypt the message. For this system to work, knowledge of the public key cannot provide any knowledge about the private key, which would compromise the cryptography system.

The RSA system generates keys that have 200+ decimal digits by using prime numbers that have 100+ decimal digits. What makes the public key so secure is that efficient methods of factoring a very large number into its prime factors is beyond current mathematical feasibility, even using today's supercomputers. Estimates of factoring such numbers range from hundreds to billions of years, essentially an impossible task.

That is not to say that a mathematical

LIFEBOAT™ Associates:
The full support software
source.

Reach for the programming horizon of the 80's with Lattice™ C, the fastest C compiler.

Set the course of your next software project towards greater power and portability by selecting Lattice C, recognized as the finest and fastest 16-bit C compiler. Lattice C, the full implementation of Kernighan and Ritchie, continues to lead the way by announcing full memory management up to 1 Megabyte. Major software houses including Microsoft, MicroPro™ and Sorcim™ are using Lattice C to develop their programs.

Lifeboat offers Lattice C with a tested set of software tools to provide a complete development system including:

HALO™	Color graphic primatives
PANEL™	Screen design aid
PMATE™	Customizable program editor
PLINK™-86	Overlay linkage editor
C-FOOD SMORGASBORD™	Screen and I/O utilities
LATTICE WINDOW™	Multi-window functions
FLOAT87™	8087 floating point math

Lattice C is available for a wide variety of 16-bit personal computers including IBM®, NCR®, Texas Instruments, Victor, Wang and other microcomputers running PC™-DOS, MS™-DOS and CP/M86.™

Call LIFEBOAT at 212-860-0300 for free information on the Lattice C family of software development tools.

LIFEBOAT
Associates

LIFEBOAT
Associates

1651 Third Avenue
New York, NY 10028
212-860-0300

Please send me free information on:

☐ **Lattice and development tools**
☐ **Corporate purchase program**
☐ **Dealer program**
☐ **OEM agreements**

☐ **Send me the complete LIFEBOAT software catalog. $3.00 enclosed for postage and handling.**

Name ____________________

Company ____________________

Address ____________________

City ____________________

State ____________ **Zip** ____________

Telephone ____________________

Circle **no. 30** on reader service card.

LATTICE, C-FOOD SMORGASBORD and LATTICE WINDOW, TM ', Lattice, Inc.
MICROPRO, TM International Corp.
SORCIM, TM Sorcim Corp.

LIFEBOAT, TM Intersoft Corp.
HALO, TM Media Cybernetics
PANEL, TM Roundhill Computer, Ltd
PMATE and PLINK, TM Phoenix Software

FLOAT87, TM Microfloat
IBM and PC, ® TM International Business Machines
MS, TM Microsoft
CP/M86, TM Digital Research

breakthrough cannot happen tomorrow and render the RSA system useless, similar to the recent breaking of the Knapsack System (a rival PKS). Knuth presents at least four methods of factoring numbers into their prime factors, including the sieve method that is used as a standard benchmark program. To factor a number, N, all we have to do is check all prime numbers less than or equal to the square root of N. If N is less than a million, there are only 168 primes to search. As numbers become larger, a prime number occurs about once every 2.3 times the number of decimal digits in the number. Thus, for 100-digit numbers, a prime number will be found once every 2300 numbers or so.

The problem then becomes: How do we find primes having 100+ decimal digits?

Large Prime Numbers

We are not really interested in factoring numbers into all their prime factors; we simply want to find a pair of prime numbers. Luckily, a theorem exists for this, although it does not say that a number is prime, only that it is not prime. The algorithm is based on Fermat's theorem, which states that if P is prime and X is not a multiple of P, then (X ** (P-1)) mod P = 1. Unfortunately, the theorem is not an if-and-only-if condition; i.e., it is not necessarily true that if the result is one, then P is prime. However, we *can* say that if the result is not one, then P is definitely not prime. (By the way, now we can see one of the reasons that we were interested in modifying the Russian Peasant algorithm to include modulo arithmetic.)

Knuth points out that some non-prime numbers can give prime testing routines great difficulty, such as (X ** 560) mod 561 = 1 for all X relatively prime to 561. To overcome this difficulty, Knuth provides a probabilistic prime test. If the algorithm is invoked K times, choosing X independently and randomly, and if the algorithm finds that the number is probably prime in all cases, then there is only one chance in (4 ** K) that the number is not prime. On the other hand, if during any pass the number is found to be non-prime by the algorithm, then the number is definitely not prime.

In Listing Six (page 40) you will find the prime test routine (PRIMTEST.RAT). It is not an implementation of Knuth's Algorithm P but is a simpler offshoot of the Fermat theorem, which we will call algorithm P'. Algorithm P' proceeds as follows:

Prime Test Algorithm

Algorithm P' (probabilistic primality test). Given an odd, positive integer, N, this algorithm attempts to decide whether or not N is prime.

P'1. [Check divisibility by 5] If the least significant digit of N is divisible by 5 (1sd(N) mod 5 = 0), then terminate the algorithm and say N is not prime.

P'2. [Generate exponent] Set M = N-1.

P'3. [Initialize] Set K = 1.

P'4. [Generate X] Let X be a random, positive integer in the range 1<X<N.

P'5. [Exponentiate] Set Y = (X ** M) mod N.

P'6. [Done?] If K = KTIMES and Y = 1, then terminate the algorithm and say N is probably prime.

P'7. [Not prime] If Y != 1, then terminate the algorithm and say N is not prime.

P'8. [Increase K] Set K = K+1 and return to Step P'4.

As with Knuth's Algorithm P, if this algorithm reports that N is not prime, then N is definitely not prime. If N is reported to be probably prime, a finite probability remains that N is not prime! I cannot say with any certainty what the probability is. I checked 561 with the above algorithm and, after fifty attempts to determine the primality of 561, the algorithm only reached five passes (K = 5) once; usually it only reached one or two passes. The algorithm could probably be made more efficient at Step P'1 by using some of the fast divisibility algorithms presented by H. T. Gordon.[5] It should be noted that KTIMES does not have to be 30, as is the case in Listing Six (see the *define* near the beginning of the listing), but reducing the value increases the probability that N will not be prime. The source code in Listing Six provides a test number count as the algorithm is executing (PUTDEC call) so that you can see how the operation is progressing.

Now that we can test for and generate large prime numbers, we are in a position to generate the public and private keys. So, let's look at the key generation procedure.

Key Generation

Before we dig into the generation of the keys, let's look at the RSA-PKS cryptography system in some detail to get a better feel for where we are heading. The RSA system is based on the use of large prime numbers, but in fact, we need only two large prime numbers, P and Q (at least 100 decimal digits each is preferable). The public (encryption) key is formed by the product of P and Q, i.e., N = P * Q. As you can imagine, factoring N is nearly impossible since N is a number composed of at least 200 decimal digits.

Next we now take the clear text and break it up into blocks; each block is formed by translating the characters into numbers and concatenating the digits to form a number, X, such that 0<X<N. To encrypt the message, we take each block, X, in turn, and form Y = (X ** 3) mod N. Each encrypted block, Y, is then transmitted to the intended receiver. The person receiving the message has a private (decryption) key, generated from P and Q. The private key, D, is formed by picking D such that D < N and (3 * D) mod ((P-1) * (Q-1)) = 1. We find that ((X ** 3) ** D) mod N = X; i.e., D is the cube root operator modulo N.

There is one further restriction on X. X must be greater than the cube root of N, otherwise we could easily decrypt the encrypted message using a standard cube root operation, since (X ** 3) mod N = X ** 3! What happens if one or more of the blocks does not satisfy this restriction? Well, we just pad the end of the block with random characters until the restriction is satisfied. Also, you might ask at this point, why can't I just take the cube root of Y and retrieve the clear text message, X? Well, think about what happened when you performed the modulo operation. The RSA-PKS operation is called a one-way, trap-door function, because it is easy to go one direction but not the other direction.

A couple of conditions on P and Q should be observed. First, (P-1) and (Q-1) cannot be divisible by 3; if (P-1) or (Q-1) is divisible by 3, then D may not decode the encrypted message because it does not produce a unique cube root. Second, (P-1) and (Q-1) should each have at least one large prime factor, otherwise the potential exists that N could be easily factored (see Knuth). Finally, P/Q should not be close to a simple fraction, or, again, N could be easily factored.

Now let's see how we can generate P and Q to get the keys. If you review the two previous paragraphs, you will agree that generating the keys is our most complex task (and the code will reflect this fact). Listing Seven (page 46) is the key generation routine (GENPKEYS.RAT). It uses every routine that we have discussed so far. Near the beginning of the program, some *defines* specify the KEYLENGTH and the BYTEMODULUS. These values are not set in concrete and can be reduced subject to some caution. BYTEMODULUS can be any value between 2 and 128, at the potential cost of increasing computation time in the prime testing routine. KEYLENGTH can be reduced at the risk of increasing the key's vulnerability. The length of 90 is equivalent to a 630-bit key length ((2 ** 7) ** 90) or a 190-decimal-digit key.

Our first task is to seed the random number generator. We must do this because the Microsoft random number routine always generates the same sequence of numbers if not seeded, thus giving everyone the same set of keys. We can enter any number from 1 to 9999999. The length of the seed is probably the main weakness of this RSA implementation. Given enough resources and the

Microsoft random number generation routine algorithm, I theoretically could go to any supercomputer or use a room full of microcomputers to generate all 10 million key sets. Then when you publish your public key, I could look it up, get the corresponding private key, and decrypt any of your messages. You could make my task more difficult by changing the value of KEYLENGTH and the associated values for the length of P and Q.

The program randomly selects the length of P and Q while assuring that their ratio is not a simple fraction. Then both P and Q are generated by the same routine, GENPRM. The algorithm used by GENPRM follows.

Generate Prime Algorithm

Algorithm GENPRM (generate a prime number - after Knuth). This algorithm generates a prime number of length Len(PRIME).

G1. [Get length of PO and K] Set Len(PO) = 2 * Len(PRIME) / 3 and Len(K) = Len(PRIME) - Len(PO). Here, PO will end up being a prime number used to generate PRIME, and K will be a multiplying factor used to generate PRIME.

G2. [Generate random PO and K] Generate a random value of PO of length Len(PO). Generate a random value of K of length Len(K).

G3. [Make PO odd and K even] If PO is even, set PO = PO + 1. If K is odd, set K = K - 1. (PO must be odd to be a prime.)

G4. [PO prime?] Test primality of PO. If PO is probably prime, go to Step G6. (Approximately Ln(PO)/2 numbers will be tested before finding a prime.)

G5. [PO not prime] Set PO = PO + 2 and repeat Step G4.

G6. [Adjust K] Find K such that K is even and K mod 3 = PO mod 3.

G7. [Get trial PRIME] Set PRIME = K * PO + 1.

G8. [PRIME prime?] Test primality of PRIME. If PRIME is probably prime, go to Step G10. (Approximately Ln(K * PO)/6 numbers will be tested before finding a prime.)

G9. [PRIME not prime] Set PRIME = PRIME + (6 * PO) and repeat Step G8. (Keeping K mod 3 = PO mod 3, K even and PRIME = K * PO + 1, implies K = K + 6 at this iteration. Substituting the new K into the equation for PRIME produces the above PRIME adjustment.)

G10. [Check PRIME + 1] Find greatest common divisor of (PRIME + 1). If the greatest prime factor of (PRIME + 1) is less than 100,000, then go to Step G9. Otherwise, terminate the

OPTIMIZING C86™
controls Charlie...

Like a Puppet on a String!

Serious programmers can use Optimizing C86 to control the IBM PC and other MS-DOS/PC-DOS computers. Features include:

- A **Full** and **Standard** implementation of the C language balancing tight control of the machine with the productivity and portability of a high level and standard language.
- **Hardware and operating system interfaces** for: graphics, interrupt control, **8087** use, I/O ports, real time applications and for producing ROMS.
- Use of the standard MS-DOS Linker and a compiler option to produce MASM Assembler output. Helps integrate with MASM and other standard languages like MS FORTRAN and PASCAL.
- **A rich set of libraries** that includes source for K&R functions, extras for string handling, graphics, sorting, floating point (8087 and 8086/88), **"Large" model** memory use (1,000K RAM), **"Small" memory** model, MS-DOS 1.1, 1.25, 2.0, 2.++.
- Support for numerous **add-on libraries** including: HALO **Graphics,** C Tools for fundamentals in many areas, PHACT for **ISAM** file management and numerous others. Ask for a list.

Pull Charlie's strings with our fast, complete, proven, reliable C Compiler—the leading compiler for serious programmers of MS-DOS and CPM-86 systems.

Computer Innovations
980 Shrewsbury Avenue
Suite J-506
Tinton Falls, N.J. 07724
(201) 542-5920

"They Say It All... We Do It All!"

Visa and MasterCard accepted.

C86 is a trademark of Computer Innovations, Inc. CPM-86 and MPM-86 are trademarks of Digital Research. MS-DOS is a trademark of Microsoft. PC-DOS is a trademark of International Business Machines.

Circle **no. 10** on reader service card.

algorithm and return PRIME.

With P and Q determined, N and D can be calculated. The public key, N, is found by N = P * Q and is of length Len(P) + Len(Q) = KEYLENGTH. The private key, D, is found by picking D such that D < N and (3 * D) mod (P-1) * (Q-1) = 1. This statement is equivalent to D = 2 * ((P-1) * (Q-1) + 1)/3. We now have our keys. After all of this trouble, we certainly want to save them so that they can be used in the encryption/decryption process.

Finally, we are ready to look at the encryption/decryption process. This process is the topic of the next section.

Encryption/Decryption

This section will be fairly short because we have already taken a look at the encryption/decryption process in the beginning of the last section. Listing Eight (page 56) depicts the encryption/decryption process. Again, the definitions of BYTEMODULUS and KEYLENGTH appear near the start of EDCRYPT.RAT. Note that they must be the same values used in the key generation process!

We start with a clear text message X = {x1, x2, . . . , xn} where 0 < xk < N. If xn, the final block, is less than the cube root of N, terminate the block with an End of File character and pad xn by concatenating random characters to the end of the block. I use ASCII control characters except ^Z, which is the CP/M End of File character. To encrypt each block, xk, we look at each character of xk and strip off the most significant bit; this ensures that 0 < char(xk) < BYTEMODULUS. Next we form yk = (xk ** 3) mod N. Finally, we scan each character of yk and set the most significant bit if it is an ASCII control character. This action prevents a ^Z or some other character from giving the RATFOR library character I/O routines a problem. The encrypted blocks Y = {y1, y2, . . . , yn} are sent to the intended receiver. Note that each encrypted block, yk, is of length Len(N).

As the authorized receiver, we can now apply our private key to decode the received message. We reblock Y into blocks of length Len(N). We take each block, yk, in turn. Each character (byte) is scanned and the most significant bit is stripped, returning the character to its original encrypted form. We now apply the inversion operation, xk = (yk ** D) mod N. The final block, xn, is scanned from the last character to the first character, looking for the End of File character to find the end of the original text. The blocks X = {x1, x2, . . . , xn} are assembled into the decrypted, clear text. EDCRYPT.RAT performs these basic operations, as well as selecting the files to encrypt/decrypt and the necessary file I/O.

Figure 1 (below) shows an example of generated keys, clear text, encrypted text, and decrypted text. The values of KEYLENGTH and KTIMES have been reduced to permit a reasonable amount of execution time: KEYLENGTH was set to 20 instead of 90 and KTIMES was set to 5 instead of 30. It took almost an hour to find the keys. The encryption of CLEARO.TXT into ENCRYPTO.TXT took about 8 minutes and the decryption of ENCRYPTO.TXT to DECRYPTO.TXT took nearly 3 hours and 19 minutes. When I used the original values of KEYLENGTH and KTIMES, I found that it took almost 11 days to find the keys.

To test how much an arithmetic coprocessor might increase performance, I used an AMD 9511 and the Redding Group APU Fortran library. Unfortunately, the library did not include a random number generator and the Microsoft module would not work with the APU library. Therefore, I had to substitute the random number routine found in Listing Nine (page 59). With these changes and using the reduced example above, I found that the routines executed in about half the time (key generation in 1:20, encryption in 0:04, and decryption in 1:33). The key generation took longer because the random number generator was not the same and different keys were produced. Encryption/decryption, however, used the same keys produced by the non-APU version to keep comparisons as close as possible. As I have pointed out before, optimizing the multiple-precision arithmetic and Russian Peasant routines could also have a great impact on the performance. However, I will leave that as an exercise for the reader.

One final topic remains. How can we sign our messages so that the receiver knows without a doubt that we sent the message? Digital signatures provide the answer.

```
Encryption Key - Len(N) & N
 20
 26 26 45 90 98 127 6 31  90 33 86 86 13 99 35 121 110 14 44 77

Decryption Key - Len(D) & D
 20
 17 60 30 60 65 127 46 49 0 90 98 46 40 95 60 13 54 79 10 83

First Prime - Len(P) & P
  7
 38 117 16 61 54 24 123

Second Prime - Len(Q) & Q
 13
 86 25 25 67 16 80 45 20 85 30 97 67 87
```

Figure 1A.

Keys and Primes generated by GENPKEYS. KEYLENGTH = 20 and KTIMES = 5.

```
This is an example of th RSA encryption/decryption algorithm. This example
uses a KEYLENGTH of 20 (140 binary digits), a P length of 49 binary digits,
and a Q length of 91 binary digits. The random number seed was 1111111.
The number of prime tests was reduced from 30 to 5. It took approximately
00:52:38 to generate the keys. It took about 00:08:02 to encrypt this file and
03:18:46 to decrypt the encrypted file.

For a RSA with a KEYLENGTH of 90 (630 binary digits), it takes approximately
11 days to generate the keys. The computer used to perform the algorithm was a
Z80-based machine running at 4Mhz under MP/M 2.0 (single user).
```

Figure 1B.

Clear (Plain) Text to be encrypted.

Digital Signatures

Now, let's consider a little "Spy vs. Spy," a la *Mad Magazine*. White 1 wants to send secure messages to White 2. However, Black is always capable of intercepting the transmitted messages and substituting his own messages. The ability of Black to intercept and substitute causes some significant problems for White, e.g., risk of dirty tricks and crosses over the eyes. The problem is: How can White 2 be sure that a received message from White 1 is the real thing and not a setup?

A plain text signature is no good, since Black could sign the message with White 1's name before encrypting the message with White 2's public key. How can White use the RSA–PKS system to confirm that a message was sent from a friend? Well, assume that there are public keys that are published in something like the phone book. Anyone is able to take his plain text message and pass it through the encryption algorithm using another's published, public key.

Let's go back to the section on modulo arithmetic and look at the commutative property of exponentiation. Eureka! We find that:

```
X = ((X ** 3) ** D) mod N
  = (((X ** 3) mod N) ** D) mod N
  = (((X ** D) mod N) ** 3) mod N
```

That is, the order of encryption and decryption algorithm application does not matter. Thus, White 1 can take his clear text message, X, and encrypt it with his private key, Dm, then encrypt the result further with White 2's public key, Ni, and finally send the resultant message, Y, to White 2.

```
Ysig = (X ** Dm) mod Nm
                         <– Signature Step
Y = (Ysig ** 3) mod Ni
                         <– Encryption Step
```

Black cannot read the message! To read the message, White 2 reverses the process. The message, Y, is passed through White 2's private algorithm, Di, and a yet garbled message is found. White 2 suspects that White 1 sent the message, so White 2 passes the resultant message through White 1's encryption algorithm, Nm, and, lo and behold, White 1's plain text message, X, appears.

```
Ysig = (Y ** Di) mod Ni
                         <– Decryption Step
X = (Ysig ** 3) mod Nm
                         <– Signature Verification Step
```

```
81 7F 6B 85 22 85 21 66  8E 40 99 3A 5A 35 8D 40   .~k.".!f.@.:Z5.@
65 9D 9D 6B 82 3C 94 74  25 8E 72 47 68 2B 50 30   e..k.<.t%.rGh+P0
68 62 6A 4D 2D 30 7A 87  8A 9C 59 28 4A 6F 5C 2C   hbjM-0z...Y(Jo\,
8E 99 20 8E 4D 45 63 28  83 7E 32 6A 98 43 2A 8D   .. .MEc(.~2j.C*.
45 70 5C 4A 61 3A 4F 63  92 6E 5F 90 44 9A 69 2E   Ep\Ja:Oc.n_.D.i.
89 82 87 9D 6F 82 6E 7F  52 61 29 4A 95 75 9A 37   ....o.n~Ra)J.u.7
49 8D 2B 70 82 85 29 98  5E 67 9E 40 25 62 9B 37   I.+p..).^g.@%b.7
9B 60 91 44 4A 73 7C 6C  83 90 4E 94 58 36 78 70   .`.DJs|l..N.X6xp
46 56 95 75 2B 82 52 7A  34 9C 46 8B 88 7A 45 3B   FV.u+.Rz4.F..zE;
59 98 38 29 8B 3C 9C 71  44 64 2A 48 7E 26 6C 7D   Y.8).<.qDd*H~&l}
8D 6B 72 3D 52 55 4B 28  4A 3A 48 6B 49 39 81 9B   .kr=RUK(J:HkI9..
43 57 42 67 8C 4C 9D 5F  58 43 2E 39 8E 34 48 36   CWBg.L._XC.9.4H6
3F 89 2F 49 97 62 86 67  93 85 42 8C 55 38 58 49   ?./I.b.g..B.U8XI
9E 46 58 78 21 6C 63 42  38 25 9C 59 8F 6C 25 7B   .FXx!lcB8%.Y.l%{
77 4E 82 33 51 65 86 98  32 88 5A 2E 83 42 6C 33   wN.3Qe..2.Z..Bl3
94 5E 56 6B 8D 33 83 51  8A 25 87 4F 99 64 99 83   .^Vk.3.Q.%.O.d..
7F 43 5A 58 8F 23 9C 8A  75 68 70 73 7F 3E 9F 80   ~CZX.#..uhps~>..
5E 75 8F 5F 3E 7B 63 43  8B 87 6F 44 83 24 5E 48   ^u._>{cC..oD.$^H
89 7F 89 81 71 5C 84 37  61 5F 63 41 8D 34 62 8C   .~..q\.7a_cA.4b.
3E 3C 83 42 58 29 64 72  56 46 60 63 7E 42 9D 79   ><.BX)drVF`c~B.y
8B 22 93 20 2A 71 37 6D  70 9E 63 99 2F 2D 61 3A   .". *q7mp.c./-a:
57 4A 85 25 80 64 87 52  39 7E 4D 38 7B 36 89 2F   WJ.%.d.R9~M8{6./
59 6F 41 6A 76 7B 91 57  8B 8B 52 64 31 92 65 4F   YoAjv{.W..Rd1.eO
50 35 90 90 91 6B 57 7C  51 87 9F 9E 97 89 40 77   P5...kW|Q.....@w
67 74 4E 2A 34 31 65 7D  3D 97 63 9C 67 61 99 52   gtN*41e}=.c.ga.R
92 8A 99 43 97 6A 4E 44  57 91 26 7F 8A 78 2A 6C   ...C.jNDW.&~.x*l
67 54 3E 6B 84 75 94 88  35 71 56 90 55 51 41 30   gT>k.u..5qV.UQA0
61 2B 8F 4D 3D 43 6C 81  89 2F 22 69 9B 5E 6A 8B   a+.M=Cl../"i.^j.
61 77 95 36 75 3E 6A 67  7F 21 5C 4C 84 63 20 93   aw.6u>jg~!\L.c .
77 7E 6D 86 21 7A 8B 59  64 3C 5E 4F 87 7F 39 29   w~m.!z.Yd<^O.~9)
8C 8C 5A 45 8C 36 9E 4E  29 61 73 63 2D 7F 7C 33   ..ZE.6.N)asc-~|3
6D 86 25 94 86 62 7F 8E  3F 44 7D 35 5B 7D 66 27   m.%..b~.?D}5[}f'
7E 7F 99 9F 71 5C 30 34  88 51 25 80 78 43 90 8D   ~~..q\04.Q%.xC..
49 8A 9D 99 48 95 94 8C  63 38 2D 2E 96 9E 26 70   I...H...c8-...&p
2B 46 4E 50 3D 7D 81 59  66 52 87 45 3B 40 26 93   +FNP=}.YfR.E;@&.
80 82 4C 8A 3E 37 4A 81  88 6D 24 2D 25 8F 65 85   ..L.>7J..m$-%.e.
20 67 45 90 97 66 7D 67  39 6F 20 23 29 46 72 7F    gE..f}g9o #)Fr~
97 85 70 47 63 9B 3B 41  8D 27 44 87 56 50 2A 41   ..pGc.;A.'D.VP*A
2B 45 45 6E 5F 74 79 44  85 2A 23 8B 91 2E 6C 89   +EEn_tyD.*#...l.
8B 4E 8F 76 8E 24 94 9F  42 85 6D 78 9A 63 48 22   .N.v.$..B.mx.cH"
92 2B 6F 5C 71 34 43 59  6A 56 96 30 47 9F 25 30   .+o\q4CYjV.0G.%0
9D 35 85 8A 87 3C 88 68  35 62 81 93 84 5D 97 61   .5...<.h5b...].a
51 7C 87 5D 67 93 66 7C  1A                        Q|.]g.f|.
```

Figure 1C.

Encrypted Text using Encryption Key in Figure 1A and Clear Text in Figure 1B.

White 2 is now sure that the message came from White 1 because no one else could know Dm! Black could not make a substitution because he cannot know White 1's private key, unless White 1 has sloppy security procedures.

To review the process, the encryption of a message using digital signatures consists of:

Y = (((X ** Dm) mod Nm) ** 3) mod Ni

The decryption process consists of:

X = (((Y ** Di) mod Ni) ** 3) mod Nm

That is all there is to it. For further information on the use of digital signatures, you should read the articles in IEEE *Computer Magazine*, February 1983.[6-9]

Summary

We have covered a lot of material in these two parts. In the first part we looked at RATFOR, modulo arithmetic, multiple-precision arithmetic, and the Russian Peasant algorithm. We saw that these routines form the core of the RSA-PKS cryptography system.

In this second part we revisited RATFOR to look at the RATDEF file and some of the RATFOR library routines. The RATDEF file contains some basic keyword and character definitions. The library routines add character I/O to the basic Fortran library.

Next, we looked at Public Key Systems and found what made them advantageous. We do not have to worry about key distribution, because the public key can be made accessible to anyone without worrying about compromising the encryption/decryption process. The public key does not provide any information about the private key.

Third, we looked at the generation of large prime numbers. We used Fermat's theorem to test odd, positive numbers for primality. The test was a probabilistic test that has a finite probability that a reported prime number is not prime, even though the probability is very small. However, if a number is said to be not prime, then it is definitely not prime.

Fourth, we looked at the generation of the public and private keys. We used all of the previously developed routines to generate the prime factors of the keys,

```
54 68 69 73 20 69 73 20   61 6E 20 65 78 61 6D 70   This is an examp
6C 65 20 6F 66 20 74 68   65 20 52 53 41 20 65 6E   le of the RSA en
63 72 79 70 74 69 6F 6E   2F 64 65 63 72 79 70 74   cryption/decrypt
69 6F 6E 0D 0A 61 6C 67   6F 72 69 74 68 6D 2E 20   ion..algorithm.
20 54 68 69 73 20 65 78   61 6D 70 6C 65 20 75 73    This example us
65 73 20 61 20 4B 45 59   4C 45 4E 47 54 48 20 6F   es a KEYLENGTH o
66 20 32 30 20 28 31 34   30 0D 0A 62 69 6E 61 72   f 20 (140..binar
79 20 64 69 67 69 74 73   29 2C 20 61 20 50 20 6C   y digits), a P l
65 6E 67 74 68 20 6F 66   20 34 39 20 62 69 6E 61   ength of 49 bina
72 79 20 64 69 67 69 74   73 2C 20 61 6E 64 0D 0A   ry digits, and..
61 20 51 20 6C 65 6E 67   74 68 20 6F 66 20 39 31   a Q length of 91
20 62 69 6E 61 72 79 20   64 69 67 69 74 73 2E 20    binary digits.
20 54 68 65 20 72 61 6E   64 6F 6D 20 6E 75 6D 62    The random numb
65 72 0D 0A 73 65 65 64   20 77 61 73 20 31 31 31   er..seed was 111
31 31 31 31 2E 20 20 54   68 65 20 6E 75 6D 62 65   1111.  The numbe
72 20 6F 66 20 70 72 69   6D 65 20 74 65 73 74 73   r of prime tests
20 77 61 73 0D 0A 72 65   64 75 63 65 64 20 66 72    was..reduced fr
6F 6D 20 33 30 20 74 6F   20 35 2E 20 20 49 74 20   om 30 to 5.  It
74 6F 6F 6B 20 61 70 70   72 6F 78 69 6D 61 74 65   took approximate
6C 79 20 30 30 3A 35 32   3A 33 38 0D 0A 74 6F 20   ly 00:52:38..to
67 65 6E 65 72 61 74 65   20 74 68 65 20 6B 65 79   generate the key
73 2E 20 20 49 74 20 74   6F 6F 6B 20 61 62 6F 75   s.  It took abou
74 20 30 30 3A 30 38 3A   30 32 20 74 6F 0D 0A 65   t 00:08:02 to..e
6E 63 72 79 70 74 20 74   68 69 73 20 66 69 6C 65   ncrypt this file
20 61 6E 64 20 30 33 3A   31 38 3A 34 36 20 74 6F    and 03:18:46 to
20 64 65 63 72 79 70 74   20 74 68 65 20 65 6E 63    decrypt the enc
72 79 70 74 65 64 0D 0A   66 69 6C 65 2E 0D 0A 0D   rypted..file....
0A 46 6F 72 20 61 20 52   53 41 20 77 69 74 68 20   .For a RSA with
61 20 4B 45 59 4C 45 4E   47 54 48 20 6F 66 20 39   a KEYLENGTH of 9
30 20 28 36 33 30 20 62   69 6E 61 72 79 20 64 69   0 (630 binary di
67 69 74 73 29 2C 20 69   74 0D 0A 74 61 6B 65 73   gits), it..takes
20 61 70 70 72 6F 78 69   6D 61 74 65 6C 79 20 31    approximately 1
31 20 64 61 79 73 20 74   6F 20 67 65 6E 65 72 61   1 days to genera
74 65 20 74 68 65 20 6B   65 79 73 2E 20 20 54 68   te the keys.  Th
65 0D 0A 63 6F 6D 70 75   74 65 72 20 75 73 65 64   e..computer used
20 74 6F 20 70 65 72 66   6F 72 6D 20 74 68 65 20    to perform the
61 6C 67 6F 72 69 74 68   6D 20 77 61 73 20 61 20   algorithm was a
5A 38 30 2D 62 61 73 65   64 0D 0A 6D 61 63 68 69   Z80-based..machi
6E 65 20 72 75 6E 6E 69   6E 67 20 61 74 20 34 4D   ne running at 4M
68 7A 20 75 6E 64 65 72   20 4D 50 2F 4D 20 32 2E   hz under MP/M 2.
30 20 28 73 69 6E 67 6C   65 20 75 73 65 72 29 2E   0 (single user).
0D 0A 1A                                            ...
```

Figure 1D.

Decrypted Text using Decryption Key in Figure 1A and Encrypted Text in Figure 1C.

USE MAGIC/L™ TO WRITE JUST ONE PROGRAM

Bet you can't stop with one.

For just $295 MAGIC/L™ is yours. And suddenly you find you've bought more than a language. MAGIC/L is an entire interactive environment; an assembler, a compiler, an editor, an I/O package, and a system call facility are all wrapped up and delivered. And now MAGIC/L is available for CP/M-based computers.

MAGIC/L is easy to learn. It has syntax similar to C and Pascal, and because it's extensible as well as interactive, it dramatically increases productivity.

Program development features include a built-in text editor, command line recall, CCP, STAT, and PIP command emulation, and the ability to store keyboard dialog on disks.

Key language features include: CHAR, INTEGER, LONG, REAL, and String data types; record structures similar to the STRUCT facility in C; and a complete I/O package that can provide random access, variable length I/O to any CP/M file.

And MAGIC/L offers great portability. Source code which runs on CP/M can be compiled unmodified and run on any other processor.

Typical applications include hardware interfacing, process control, games creation, interactive graphics and image processing. MAGIC/L has made programming easier for DEC, 68000, and Data General users. Now it's working for CP/M users too.

MAGIC/L provides everything you need to write a complete program. But the only way to be convinced is to try it yourself. Send us your $295 check or money order—we also accept MasterCard and VISA—we'll send MAGIC/L for CP/M to you at once. A full money back guarantee is part of the package. Once you've sampled that first program, you'll have to try another . . . and another . . . and another. MAGIC/L . . . it's more than a language.

MAGIC/L . . . It's more than a language

LOKI ENGINEERING, INC.
55 Wheeler St., Cambridge, MA 02138 (617) 576-0666

Circle **no. 31** on reader service card.

P and Q. These prime numbers were then used to generate the public and private keys, N and D, respectively.

Next, we used the keys to encrypt and decrypt plain text messages. We saw how to block out the messages to encode or decode them. We found that we must pad the last block if it is too short to keep someone from easily decrypting it. Also, we found that using an APU can halve the time required to perform the encryption/decryption process.

Finally, we looked at digital signatures. We saw that another person could sign a message digitally by encrypting the message initially with their private key and then encrypting the result with the intended receiver's public key. The receiver reverses the process to confirm that the message was sent by the signee.

I hope you have enjoyed this article and found it understandable and educational. I owe Donald Knuth most of the credit for this article, since his book was the major contributor to my understanding of the RSA-PKS cryptography system. Of course, this article would not have been possible without the basic papers of Diffie and Hellman and of Rivest, Shamir, and Adleman. As I stated previously, I am sure there are better ways to implement the algorithms and even better and more efficient algorithms that could be used to improve the performance. I invite you to improve any part of what I have presented.

References

[1] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Trans Info Theory*, Vol. IT-22, No. 6, pp. 644-654.

[2] M. E. Hellman, "The Mathematics of Public Key Cryptography," *Sci Amer*, February 1979, Vol. 241, No. 2, pp. 146-157.

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures & Public Key Cryptosystems," *Comm ACM*, February 1978, Vol. 21, No. 2, pp. 120-126.

[4] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 2nd ed, Reading, MA: Addison-Wesley, pp. 250-268, 374-380, 386-389, 442-443.

[5] H. T. Gordon, "Fast Divisibility Algorithms," *DDJ*, June 1983, No. 80, pp. 14-16.

[6] S. G. Akl, " Digital Signatures: A Tutorial Survey," *Computer*, February 1983, Vol. 16, No. 2, pp. 15-24.

[7] D. W. Davies, "Applying the RSA Digital Signature to Electronic Mail," *Computer*, February 1983, Vol. 16, No. 2, pp. 55-62.

[8] R. DeMillo and M. Merritt, "Protocols for Data Security," *Computer*, February 1983, Vol. 16, No. 2, pp. 39-50.

[9] D. E. Denning, "Protecting Public Keys & Signature Keys," *Computer*, February 1983, Vol. 16, No. 2, pp. 27-35.

Additional References

[1] B. Schanning, "Securing Data Inexpensively Via Public Keys," *Computer Design*, April 5, 1983, pp. 105-108.

[2] J. Smith, "Public Key Cryptography," *Byte*, January 1983, pp. 198-218. DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 194.**

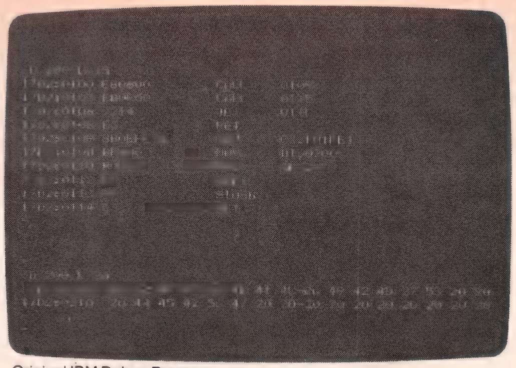
RSA (Text begins on page 30)

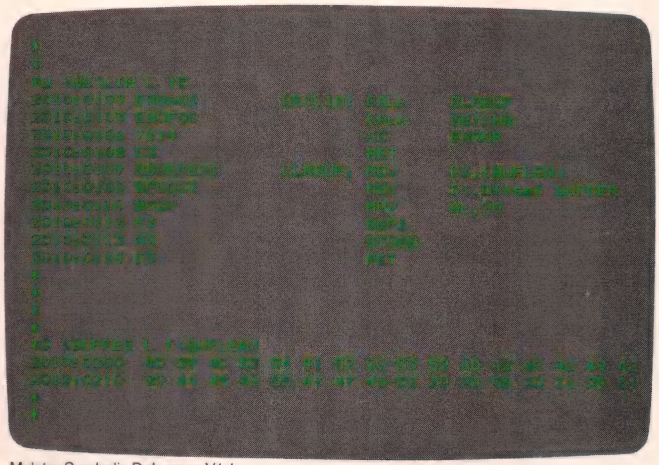
Listing Six

```
# PROGRAM NAME:  PRIMTEST.RAT
# PURPOSE:  This routine is a probabilistic test for the primality of a number.
#           If the test is true, there is a probability of (1/4)**K that the
#           number is not prime.  The test is based on Fermat's Theorem.
#           (re. D.E. Knuth, The Art of Computer Programming, V. 2
#           (Semi-Numerical Algorithms), 2nd Ed.,(Addison-Wesley, Reading, MA),
#           pp. 374-380.)
#
# LANGUAGE:  RATFOR
# AUTHOR:  CEB
# USAGE:  LOGCL=PRMTST(PRIMEQ,LENP,WORK,LENW,MODULO)
#              <PRMTST -- Logical function, indicates that the number in
#                         question (PRIMEQ) is probably prime with probability
#                         1-(1/4)**K.  PRMTST = .TRUE. if probably prime,
#                         and .FALSE. if it is definately not prime.
#              >PRIMEQ -- Byte array, contains the number (byte modulus MODULO)
#                         to be tested.  The MSDigit(s) are in PRIMEQ(1) and
#                         the LSDigit(s) are in PRIMEQ(LENP).
#                         CAUTION:  PRIMEQ > 2
#              >LENP -- Integer variable, defines length of PRIMEQ array.
#              <WORK -- Byte array, a working array needed to do test.
#              >LENW -- Integer variable, defines length of WORK array.
#                       CAUTION:  LENW = 7 * LENP + 3
#              >MODULO -- Integer variable, defines the arithmetic modulus that
#                         is to be used.  MODULO has a byte-wide effect and
#                         should be between 2 and 128 (e.g. 100 for Decimal
#                         Numbers and 128 for ASCII Characters).
#
```

(Continued on page 42)

The difference


Original IBM Debug Program


Mylstar Symbolic Debugger V1.1

is Mylstar's Symbolic Debugging Program*

The plain and simple difference is that Mylstar's Symbolic Debugging Program speaks to your IBM PC in a language you both can understand, plain and simple.

Employing the same command structure, it allows you to use symbol names, mathematical expressions, batch files, on-line help, multi-command macros and other time-saving entries.

It's the enhancement to the *IBM Debug Program* you've been looking for—because it fills in the gaps—shortening the frustrating debugging process by as much as 50%—leaving you more time to do the work you need to do and the work you want to do, plain and simple.

Mylstar's Symbolic Debugging Program has been programmer-tested for over a year at Mylstar Electronics, Inc., (formerly D. Gottlieb & Co.), designers of the video arcade game, Q*BERT™.

TO ORDER...
Call (312) 562-7400 or mail coupon today.

**Designed for IBM PC-DOS 1.1 with 128K RAM minimum*

MYLSTAR ELECTRONICS INC.

165 West Lake Street
Northlake, Illinois 60164

A Columbia Pictures Industries Company

Mylstar Electronics, Inc.,
165 W. Lake St., Northlake, IL 60164

Please send me Mylstar's Symbolic Debugging Program for use with the IBM PC computer. Enclosed is $125, plain and simple.

☐ Check ☐ Money Order 118

NAME ______________________

FIRM ______________________

ADDRESS ______________________

CITY ______________ STATE ________ ZIP ________

Illinois residents add 7% sales tax
Allow 2-4 weeks for delivery

Circle **no. 39** on reader service card.

RSA (Listing Continued, text begins on page 30)

Listing Six

```
# ARRAYS USED: PRIMEQ(*)
# EXTERNALS:  RPEXP,MPSUBT
# UPDATE HISTORY:  INITIAL RELEASE -- 01/25/83 CEB
#
define(KTIMES,30) # number of times, K, to test PRIMEQ

logical function prmtst(primeq,lenp,work,lenw,modulo)

      character back3(4)
      byte primeq(1),work(1),one,notprm
      logical xflag
      integer lenp,lenw,modulo

      data back3/BACKSPACE,BACKSPACE,BACKSPACE,EOS/

      data one/1/
      data rndnxt/.5/ # make RAN(*) generate the next random number
                      #  (any positive number will do)
                      #  *** could seed it with a prior call to RAN(-.xxxxxx)
                      #      where <xxxxxx> is entered by the user ***

      prmtst=.false. # initialize to indicate not prime
      #######
      #
      # The following test assumes PRIMEQ != 5 (i.e. the prime number 5 is not
      #  sought).  If assumption is invalid them remove this test.

      lsdig=primeq(lenp) # get the LSDigit of PRIMEQ
      if ((modulo > 5) & (mod(lsdig,5) == 0)) # PRIMEQ divisible by 5?
           return # no need to check further so Exit !!!


      #
      #######
      idxprm=1 # initialize pointer to 1st digit of PRIMEQ
      while (primeq(idxprm) == 0 & idxprm <= lenp) # digit is zero
            idxprm=idxprm+1 # move index to next position
      lenp0=lenp-idxprm+1 # get length of right justified prime
      if ((lenw >= 7*lenp0+3) & (lenp0 >= lenp) &
           ((lenp0 != 1) | (primeq(1) > 2))) # work array has sufficient length
                                              #  and PRIMEQ > 2?
            {
            idxexp=lenp0+1 # get index to exponent
            call mpsubt(primeq(idxprm),lenp0,one,1,work(idxexp),lenp0,modulo)
                                            # generate (PRIMEQ - 1), i.e. exponent
            idxrlt=idxexp+lenp0 # get index to RESULT: X ** (PRIMEQ-1) mod PRIMEQ
            idxscr=idxrlt+lenp0 # get index to scratch work area for calculations
            fmodul=float(modulo) # float MODULO
            notprm=NO # initialize Not Prime flag
            call remark('      Test Number:  .') # indicate test running
            do i=1,KTIMES # test for primality
                  {
                  call putdec(i,3) # show test number
                  call remark(back3) # backspace 3
                  idxx=1 # initialize index to X
                  idigit=primeq(idxprm) # get 1st digit of PRIMEQ
                  if (idigit == 1) # first digit a 1?
                        {
                        xflag=.true. # indicate 1st digit of X is zero
                        ix=0 # force X < PRIMEQ
                        }
                  else # first digit > 1
                        {
```

(Continued on page 45)

GTEK INC.

DEVELOPMENT HARDWARE/SOFTWARE

HIGH PERFORMANCE/ COST RATIO

(601) 467-8048

EPROM PROGRAMMER

Model 7128-L1,L2,L2A	$239.00
Model 7128-24	$329.00
DR8 or DR5	$ 30.00
DR8PGX or DR5PGX	$ 75.00
Cross Assemblers	$200.00
XASM (for MSDOS)	$250.00
U/V Eraser DE-4	$ 78.00
RS232 Cables	$ 30.00
8751 adapter	$174.00
8755 adapter	$135.00
48 Family adapter	$ 98.00

Compatible w/all Rs 232 serial interface port * Auto select baud rate * With or without handshaking * Bidirectional Xon/Xoff and CTS/DTR supported * Read pin compatible ROMS * No personality modules * Intel, Motorola, MCS86, Hex formats * Split facility for 16 bit data paths * Read, program, formatted list commands * Interrupt driven, program and verify real time while sending data * Program single byte, block, or whole EPROM * Intelligent diagnostics discern bad and erasable EPROM * Verify erasure and compare commands * Busy light * Complete w/Textool zero insertion force socket and integral 120 VAC power (240 VAC/50Hz available)

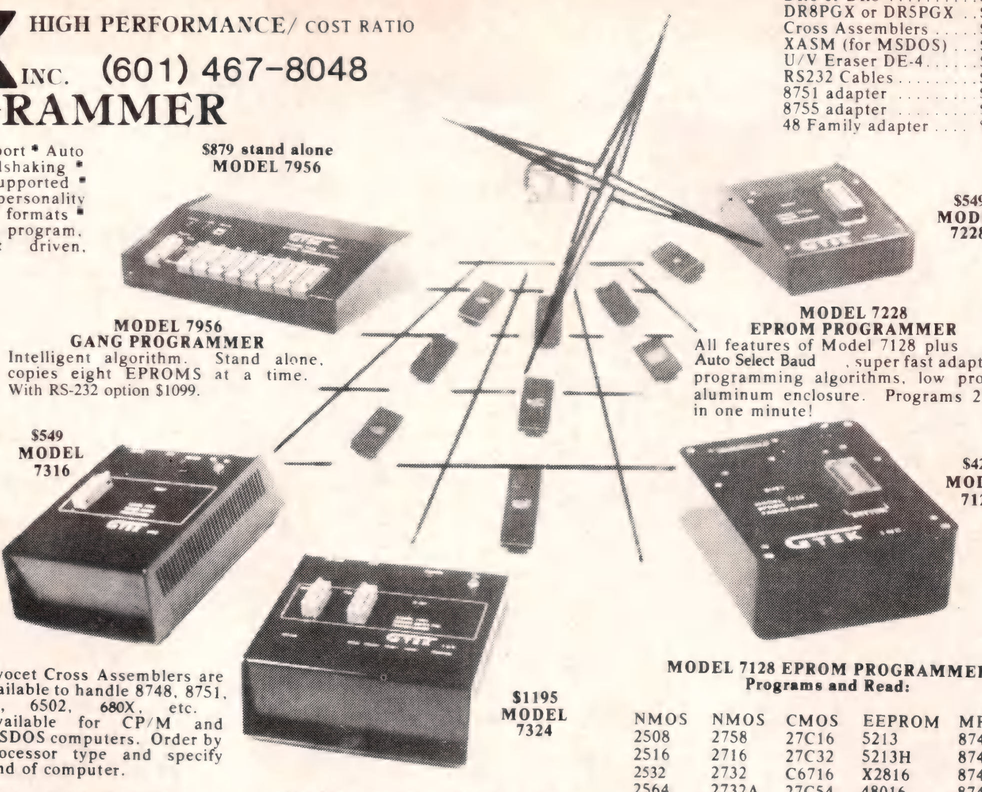
DR Utility Package allows communication with 7128, 7228, and 7956 programmers from the CP/M command line. Source Code is provided. PGX utility package allows the same thing, but will also allow you to specify a range of addresses to send to the programmer, Verify, set the Eprom type.

MODEL 7316 PAL PROGRAMMER
Programs all series 20 PALS. Software included for compiling PAL source codes.

Software Available for CPM.[1] ISIS.[2] TRSDOS.[3] MSDOS.[4]

1. TM of Digital Research Corp.
2. TM of Intel Corp.
3. TM of Tandy Corp.
4. TM of Microsoft.

Post Office Box 289
Waveland, Mississippi 39576
[601]-467-8048



$879 stand alone
MODEL 7956

MODEL 7956
GANG PROGRAMMER
Intelligent algorithm. Stand alone, copies eight EPROMS at a time. With RS-232 option $1099.

$549
MODEL
7228

MODEL 7228
EPROM PROGRAMMER
All features of Model 7128 plus Auto Select Baud , super fast adaptive programming algorithms, low profile aluminum enclosure. Programs 2764 in one minute!

$549
MODEL
7316

$429
MODEL
7128

$1195
MODEL
7324

Avocet Cross Assemblers are available to handle 8748, 8751, Z8, 6502, 680X, etc. Available for CP/M and MSDOS computers. Order by processor type and specify kind of computer.

Model DE-4 U/V Products hold 8, 28 pin parts. High quality professional construction.

MODEL 7324 PAL PROGRAMMER
Programs all series 20 & 24 PALS. Operates stand alone or via RS232.

MODEL 7128 EPROM PROGRAMMER
Programs and Read:

NMOS	NMOS	CMOS	EEPROM	MPU'S
2508	2758	27C16	5213	8748
2516	2716	27C32	5213H	8748H
2532	2732	C6716	X2816	8749H
2564	2732A	27C54	48016	8741
68766	2764		12816A	8742H
68764	27128			8741H
8755	27256			8751
5133				

Circle **no. 23** on reader service card.

LET YOUR MICRO DO THE DIRTY WORK...

SO *YOU* CAN CONCENTRATE ON PROBLEM-SOLVING
Handle more data and solve more complex problems, with

CALCULUS FOR SCIENTISTS AND ENGINEERS
Produced by SCIENTIFIC SOFTWARE, INC.

Computer software that performs all the functions of calculus!
Calculus does single, double, and triple integrations; nth order systems of ordinary differential equations; and locates minima, maxima, and zeros of a function. Its innovative algorithms minimize the lengthy execution time associated with programmable calculators or software books, where programs have to be keyed in.

A program that's as flexible as you need it to be
Calculus quickly plots and tabulates a function and its derivatives, and then displays *both* on the screen or in hard copy printout, *the way you want to see it.*

And it's easier to use than your calculator.
By using menus and prompts, you can choose when and where to change your inputs without prior preparation. As fresh numerical data appear with each successive iteration, you have the option to see results of earlier iterations. You can *stop* consecutive iterations as soon as you're within reach of a specified accuracy.

Name any calculus problem— *Calculus* provides the answer.

CALCULUS FOR SCIENTISTS AND ENGINEERS
comes in two versions, for the Apple® II/IIe with 48K memory and for the IBM PC with 64K memory.
Apple II/IIe version: (0 471 88205-4) $125.00
IBM PC version: (0 471 88048-5) $125.00

Look for Wiley Professional Software at your local computer store or bookshop. For faster service

CALL TOLL FREE:
1 800 526-5368
In New Jersey, call collect (201) 342-6707
Order # 4-1718
VISA, MasterCard, and American Express accepted on phone orders.

Wiley Professional Software
a division of John Wiley & Sons, Inc.
605 Third Avenue New York, N.Y. 10158

4-1722

PROFESSIONAL SOFTWARE

Unlocking the power of computing

Circle **no. 66** on reader service card.

Don't call her cheap. Call her beautiful.

The Bonnie Blue™

Word Processing System for the IBM Personal Computer

It's obvious what makes her so cheap, but what makes Bonnie Blue so beautiful? Bonnie Blue is a new and easy-to-use word processing program for the IBM Personal Computer.

The Full System. The Bonnie Blue System includes in one program a full screen Editor, a Printing module and a useful Toolbox. It includes the features you've come to expect, and more:

complete cursor control: by character, word, line; page up and down instantly; go to top, bottom of document; auto scroll towards top or bottom
word wrap
margin justification, centering
adjustable margins, tabs, indents
reformat paragraphs
move, copy, delete, paste blocks
find with delete, insert, replace and wild card characters
keyboard remapping
multi-line headers, footers

Bonnie Blue can handle lines longer than the screen is wide, by horizontally scrolling the line. And, unlike some programs, Bonnie Blue lets you include any displayable character in your text, such as block graphics and foreign language characters.

Unique Features. With Bonnie Blue, you can "paint" display attributes onto your text, by the character, word, or line, or automatically as you enter text. With the monochrome adapter, you can paint any combination of underlined, bold, reverse video or blinking. With an 80 column monitor and the color/graphics adapter, this translates into a palette of 16 color combinations to choose from. And if your computer has both monitors, Bonnie Blue lets you use them both, shifting back and forth as you wish.

Powerful Printing Module. You can use these colors or display attributes to highlight text on the screen, and Bonnie Blue can remove them from a file when you want (all files created by Bonnie Blue are DOS standard). The Printing module understands these text attributes, and you can map them into any single printer function or combination.

For example, normally you would want underlined text to print underlined. But you can tell Bonnie Blue to print underlined characters as both underlined and bold. Bright text on the screen can mean double struck, or emphasized and in italics. You are at the controls.

The first Print formatting module supports all the text capabilities of the Epson MX series with Graftrax Plus. By the time this ad appears, we will be supporting other popular dot-matrix and letter quality printers.

More than thirty "dot" commands give you added control of the format of your finished document. You can send it to a disk file instead of the printer, or preview the final page formatting on the screen.

Toolbox. The Toolbox is a set of useful functions, called "filters" that allow you to extract information from your files and transform their content. With these tools, you can join files together, sort lines of text, count words, find and substitute patterns, etc. Writers and programmers find this a useful collection of productivity enhancers.

Bonnie Blue is also great for a hard disk system. A thorough User's Guide, complemented by help screens and roadmaps, make the Bonnie Blue an exceptionally easy-to-learn and easy-to-use system.

Order yours today, or send for our free brochure. Bonnie Blue is available exclusively from **Bonnie Blue Software,** P.O. Box 536, Liverpool, NY 13088.

IBM Personal Computer is a trademark of IBM Corp. Epson Graftrax Plus is a trademark of Epson America Inc.

Only $50

Bonnie Blue Software
Post Office Box 536
Liverpool, NY 13088

☐ Send me the Bonnie Blue System. I am enclosing $50 (NY State residents please add 7% sales tax).

☐ Please send literature.
I have a ____________

☐ Check enclosed ☐ VISA ☐ MasterCard Sorry, no COD.

Credit Card No.____________________ Expires__________

Signature______________________________

Name______________________________

Address______________________________

City____________________ State__________ Zip__________

Company______________________________

Minimum recommended system:
IBM PC, 128K, 2 disk drives, PC-DOS 1.1 or 2.0, 80-column monitor or monochrome adapter, or both, Epson MX-80 or MX-100 with Graftrax Plus.

Versions available soon for PCjr. Write for details.

Circle **no. 6** on reader service card.

Listing Six

```
          xflag=.false. # indicate 1st digit of X is non-zero
          repeat # get first digit of X
               {
               ix=mod(int(fmodul*ran(rndnxt)),idigit)
                                                    # force X < PRIMEQ
               }
          until  ((lenp0 != 1 & ix != 0) | (lenp0 == 1 & ix > 1))
                                   # 1st digit of X is valid
          }
     work(idxx)=ix # initialize 1st digit of X
     idxx=idxx+1 # move index to next position
     while (idxx <= lenp0) # generate a random integer
                         #  1 < X < PRIMEQ
          {
          if (xflag) # second digit of X needs special handling?
               {
               xflag=.false. # indicate 2nd digit of X is non-zero
               repeat # get second digit of X
                    {
                    ix=mod(int(fmodul*ran(rndnxt)),modulo)
                                                    # force X >= 2
                    }
               until  ((lenp0 != 2 & ix != 0) | (lenp0 == 2 & ix > 1))
                                        # 2nd digit of X is valid
               }
          else # PRIMEQ < X < 1

               ix=mod(int(fmodul*ran(rndnxt)),modulo) # get next
                                                        #  digit
          work(idxx)=ix # load next digit of X
          idxx=idxx+1 # move index to next position
          }
     call rpexp(work,lenp0,work(idxexp),lenp0,work(idxrlt),lenp0,
               primeq(idxprm),lenp0,work(idxscr),4*lenp0+3,modulo)
                    # generate RESULT = X ** (PRIMEQ -1) mod PRIMEQ
     #
     #  RESULT = (X ** (PRIMEQ-1) mod PRIMEQ) != 1 --> not prime
     #
     idxwrk=idxscr-2 # get index to Digit above LSDigit of RESULT
     if (work(idxwrk+1) != 1) # LSDigit of RESULT --> not prime?
          notprm=YES # indicate Not Prime
     else # check rest of Digits
          {
          while (idxwrk >= idxrlt) # check the rest of RESULT's
                                   #  Digits
               {
               if (work(idxwrk) != 0) # not prime?
                    {
                    notprm=YES # indicate Not Prime
                    break # leave WHILE loop
                    }
               idxwrk=idxwrk-1 # move index to next position
               }
          }
          if (notprm == YES) # not prime?
               break # leave DO loop
          }
     if (notprm == NO) # high probability PRIMEQ is a prime?
          prmtst=.true. # indicate probably prime
     }
else # cannot test PRIMEQ
     {
```

(Continued on next page)

RSA (Listing Continued, text begins on page 30)

Listing Six

```
          call rmrkln('.')
          call error('Length of [WORK] too small or [PRIMEQ] <= 2 !!!.')
                                              # print message and exit
          }
     return

end
```

End Listing Six

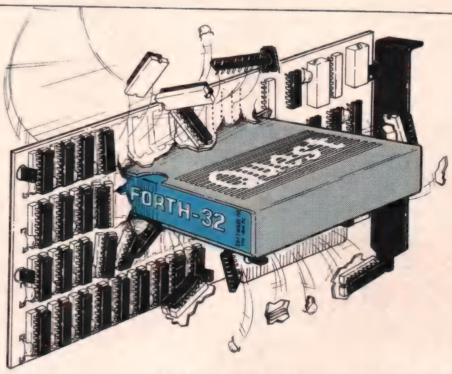
Listing Seven

Copyright © 1983, Charles E. Burton, Denver, Colorado.
All rights reserved. Permission granted to use this software for personal, noncommercial purposes only.

```
# PROGRAM NAME:  GENPKEYS.RAT
# PURPOSE:  This routine generates the public keys for the RSA
#           (Rivest-Shamir-Adleman) Public Key Encryption/Decryption System.
#           It uses the method described by D.E. Knuth, The Art of Computer
#           Programming, V. 2 (Semi-Numerical Algorithms), 2nd Ed., (Addison-
#           Wesley, Reading, MA), pp. 386-389.
#
# LANGUAGE:  RATFOR
# AUTHOR:  CEB
# USAGE:  ******** (main program)
#
#
#
# ARRAYS USED:  P(KEYLENGTH),Q(KEYLENGTH),N(KEYLENGTH),D(KEYLENGTH),
#               WORK(WORKLENGTH)
# EXTERNALS:  MPADD,MPSUBT,MPMULT,MPDIV,RPEXP,PRMTST
# UPDATE HISTORY:  INITIAL RELEASE -- 01/28/83 CEB
#

define(KEYLENGTH,90) # length of key for characters (assume 80 char. lines)
define(WORKLENGTH,810) # length of WORK buffer for characters
                       #  (9 * KEYLENGTH)
define(BYTEMODULUS,128) # byte modulus for characters (1 character/byte)

define(LUNOUT,6) # output logical unit

program pubkey

     character type
     byte p(KEYLENGTH),q(KEYLENGTH),n(KEYLENGTH),d(KEYLENGTH),work(WORKLENGTH),
          one,two,three
     double precision seed # allow entry of up to 8 characters

     equivalence (seed,work(1))

     data one,two,three/1,2,3/
     data rndnxt/0.5/ # positive number needed to generate next random number
     data modulo/BYTEMODULUS/ # get byte modulus for arithmetic


     write(CONSOLE,100)
100  format(1x,'Enter a seed for the random number generator (XXXXXXX):  ')
     read(CONSOLE,200) seed
200  format(a7)
     rnd1st=0.0 # initialize random number seed
```

```
do i=1,7 # generate random number generator seed
      {
      if (type(work(i)) == DIGIT) # entry is a digit?
            {
            inum=work(i)-DIG0 # get value of seed digit
            rnd1st=10.0*rnd1st+float(inum) # add digit to seed
            }
      else # non-digit found
            break # exit DO loop
      }
rnd1st=-rnd1st/1.0e7 # -1.0 < seed <= 0
call ran(rnd1st) # seed random number generator
len=KEYLENGTH # get key length desired (i.e. length of P * Q)
lendel=4+int(3.0*ran(rndnxt)) # get a delta length between 4 and 7
                                       #  (i.e. P and Q length diff. of 8 and 12)
lenp=len/2-lendel # get desired length of prime factor P
lenq=len-lenp # get desired length of prime factor Q
call rmrkln('.')
call rmrkln('Generating P.')
call genprm(p,lenp,work,modulo) # generate prime factor P
call rmrkln('.'); call rmrkln('.')
call rmrkln('Generating Q.')
call genprm(q,lenq,work,modulo) # generate prime factor Q
#
# generate the Public Key encryptor:
#    N = P * Q
#
call rmrkln('.'); call rmrkln('.')
call rmrkln('Generating N.')
call mpmult(p,lenp,q,lenq,n,len,modulo)
idx1=1 # get index to 1st digit of N
```

(Continued on next page)



Break Through the 64K Barrier!

FORTH-32™ lets you use up to one megabyte of memory for programming. A Complete Development System! Fully Compatible Software and 8087 Floating Point Extensions.

Quest™
Quest Research, Inc.
303 Williams Ave.
Huntsville, AL 35801
(205) 533-9405

Call today toll-free or contact a participating Computerland store.

800-558-8088

Now available for the IBM PC, PC-XT, COMPAQ, COLUMBIA MPC, and other PC compatibles!

IBM, COMPAQ, MPC, and FORTH-32 are trademarks of IBM, COMPAQ, Columbia Data Products, and Quest Research, respectively.

Circle **no. 48** on reader service card.

RSA (Listing Continued, text begins on page 30)

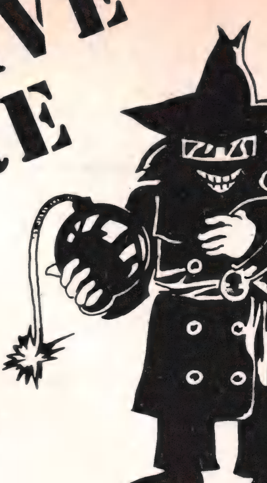
Listing Seven

```
while (n(idx1) == 0) # find 1st non-zero digit of N
     idx1=idx1+1 # advance to next position
if (idx1 > 1) # need to shift N to make 1st digit non-zero (since modulus
              #  of exponential operation cannot have a leading zero)?
     {
     len=len-(idx1-1) # adjust length of N
     do idx2=1,len # left justify N, stripping zeroes
          {
          idx3=idx1+idx2-1 # get index to digit to move
          n(idx2)=n(idx3) # left justify N
          }
     }
#
# generate Public Key decryptor:
#   D = 2 * ((P - 1) * (Q - 1) + 1) / 3 = 2 * (N - P - Q) / 3 + 1
#
call rmrkln('.')
call rmrkln('Generating D.')
len1=max0(lenp,lenq)+1 # get length of (P + Q)
call mpadd(p,lenp,q,lenq,work,len1,modulo) # generate (P + Q)
idx1=len1+1 # get index to result of N - (P + Q)
call mpsubt(n,len,work,len1,work(idx1),len,modulo) # generate N - (P + Q)
idx2=idx1+len # get index to start of result of 2 * (N - P - Q)
len2=len+1 # get length of result of 2 * (N - P - Q)
     call mpmult(work(idx1),len,two,1,work(idx2),len2,modulo)
                                                # generate 2 * (N - P - Q)
     work(1)=0 # insure Numerator starts with 0 for division normalization
     do idx=1,len2 # copy Numerator <2 * (N - P - Q)>
          {
          work(idx+1)=work(idx2)
          idx2=idx2+1 # advance index to next position
          }
     len1=len2+1 # account for extra digit in numerator
     idx2=len1+1 # get index to quotient of <2 * (N - P - Q) / 3>
     len2=len1-1 # get length of quotient
     idx3=idx2+len2 # get index to remainder of <2 * (N - P - Q) / 3>
     call mpdiv(work,len1,three,1,work(idx2),len2,work(idx3),1,modulo)
                                 # generate 2 * (N - P - Q) / 3 (i.e. Quotient)
     len3=len2+1 # get length of result of 2 * (N - P - Q) / 3 + 1
     call mpadd(work(idx2),len2,one,1,work(idx3),len3,modulo)
                                 # generate 2 * (N - P - Q) / 3 + 1
     do idx=1,len # copy 2 * (N - P - Q) / 3 + 1 to D from LSDigit to MSDigit
                  #  Note: D < N, so Length(D) = Length(N)
                  #        LEN3 = LEN + 3
          {
          idx1=len-idx+1 # get index to D
          idx2=idx3+len3-idx # get index to 2 * (N - P - Q) / 3 + 1
          d(idx1)=work(idx2) # move 2 * (N - P - Q) / 3 + 1 to D
          }
     # send Keys to CONSOLE
     call rmrkln('.')
     call rmrkln('Keys and Key Factors.')
     write(CONSOLE,300) len # N
300  format(1x,20(i3,1x))
     write(CONSOLE,300) (n(idx), idx=1,len)
     write(CONSOLE,300) len # D
     write(CONSOLE,300) (d(idx), idx=1,len)
     write(CONSOLE,300) lenp # P
     write(CONSOLE,300) (p(idx), idx=1,lenp)
     write(CONSOLE,300) lenq # Q
     write(CONSOLE,300) (q(idx), idx=1,lenq)
```

(Continued on page 50)

SUBVERSIVE SOFTWARE

So cheap and useful it's...dangerous!



A radical idea in software development. Useful Pascal programs, with:

- Complete annotated *source listings;*
- Disk(s) with *source code and compiled code;*
- User manual;
- Complete programmer documentation describing data structures and algorithms; and giving suggestions for modification.

Modify them, include them in your own systems, or simply use them.

A growing library of software tools you'll find hard to resist.

SUBVERSIVE SOFTWARE

TPL

The Text Processing Language. A text-file runoff program consisting of a set of text-processing primitive commands from which more complex commands (macros) can be built (as in Logo). Features include:

- Complete customization of text processing through macro definition and expansion, looping structures, and conditional statements;
- Adapts to any printer;
- Pagination;
- Text justification and centering;
- Indexing and tables of contents;
- Superscripts and subscripts;
- Bolding and underlining;
- Multiple headers and footers;
- End notes and footnotes;
- Widow and orphan suppression;
- Floating tables and 'keeps.'

$50

SUBVERSIVE SOFTWARE

DBX

Blocked Keyed Data Access Module. Maintains disk files of keyed data. Can be used for bibliographies, glossaries, multi-key data base construction, and many other applications.

- Variable-length keys;
- Variable-length data;
- Sequential access and rapid keyed access;
- Single disk access per operation (store, find, delete) in most cases;
- Multiple files;
- Dynamic memory allocation for RAM-resident index and current "page" of entries;
- Includes demonstration program and testbed program.

$50

SUBVERSIVE SOFTWARE

PDMS

The Pascal Data Management System. A user-oriented data management system in which numeric and alphanumeric data are stored in tables with named columns and numbered rows. Currently being used for dozens of different kinds of business and scientific applications, from inventory management to laboratory data analysis. Includes over 20 Pascal programs; more than 10,000 lines of code. Main features include:

- Maximum of 32,767 rows per file;
- Maximum of 400 characters per row, and 40 columns per table;
- Full-screen editing of rows and columns, with scrolling, windowing, global search/replace, and other editing features;
- Sorting, copying, merging, and reducing routines;
- Mailing label program;
- Reporting program generates reports with control breaks, totals and subtotals, and selects rows by field value; many other reporting features;
- Cross-tabulation, correlations, and multiple regression;
- Video-display-handling module;
- Disk-file-handling module.

Many other features. UCSD formats only.

$250

SUBVERSIVE SOFTWARE

ZED

Full-screen text editor; designed to be used either with TPL or by itself.

- Full cursor control;
- Insert mode with word wrap;
- 'Paint' mode;
- Single-keystroke or dual-keystroke commands;
- Command synonyms;
- Global search and replace;
- Block move, block copy, and block delete.

$50

BVERSIVE FTWARE

SCINTILLA

A log logit curve fitting program for radio-immunologic data; must be used with PDMS (described above).

- Multiple protocol files;
- Quality control files;
- Four-parameter non-linear curve fit.

UCSD formats only. **$250**

SUBVERSIVE SOFTWARE

CHROME

Chromatography data analysis program:

- Graphic display of analog data;
- Panning and zooming;
- Automatic peak-finding and baseline calculation;
- Full interactive peak editing;
- Computation of peak areas;
- Strip charts on C. Itoh and EPSON printers.

$100

SUBVERSIVE SOFTWARE

PLANE

Planimetry program:

- Bit-pad entry of cross sections;
- Real-time turtlegraphics display;
- Calculation of areas;
- Saves calculations to text file.

$100

VERSIVE TWARE

MINT

A terminal emulation program for communication between computers of any size.

- User-configurable uploading and downloading of files;
- X-ON/X-OFF and EOB/ACK protocols;
- Interrupt-driven serial input (for Prometheus Versacard in Apple II);
- Printer-logging.

$50

For more information, call 919-942-1411. To order, use form below or call our toll-free number: 1-800-XPASCAL

Check appropriate boxes:

FORMAT	PRODUCT	PRICE
☐ 8″ UCSD SSSD	☐ DBX	$ 50
☐ 5¼″ Apple Pascal	☐ PDMS	$250
☐ 5¼″ UCSD IBM PC 320k	☐ TPL	$ 50
☐ 8″ CP/M SSSD	☐ ZED	$ 50
☐ 5¼″ IBM MS-DOS	☐ MINT	$ 50
☐ 5¼″ CP/M Osborne	☐ SCINTILLA	$250
	☐ CHROME	$100
	☐ PLANE	$100

Name ______________________________

Address ______________________________

☐ MasterCard ☐ VISA ☐ Check ☐ C.O.D.

(Please include card # and expiration date)

SUBVERSIVE SOFTWARE A division of Pascal & Associates, 135 East Rosemary St., Chapel Hill, NC 27514

Circle **no. 42** on reader service card.

Listing Seven

```
     # send Keys to LUNOUT file
     call open(LUNOUT,'keys    dat',0) # open Key file on current drive
     write(LUNOUT,400) len # N
400  format(20(i3,1x))
     write(LUNOUT,400) (n(idx), idx=1,len)
     write(LUNOUT,400) len # D
     write(LUNOUT,400) (d(idx), idx=1,len)
     write(LUNOUT,400) lenp # P
     write(LUNOUT,400) (p(idx), idx=1,lenp)
     write(LUNOUT,400) lenq # Q
     write(LUNOUT,400) (q(idx), idx=1,lenq)
     endfile LUNOUT

end
##########################################################################

# GENPRM -- Generate a prime number

subroutine genprm(prime,lenp,work,modulo)

     byte prime(1),work(1),one,two,three,four,six,p0mod3,kadj
     logical prmtst
     integer lenp,modulo

     data one,two,three,four,six/1,2,3,4,6/

     len1=(2*lenp)/3 # get length of prime number needed to generate P or Q
     len2=lenp-len1 # get length of random number needed to generate P or Q
     call rmrkln('     Generating P0.')
     call genrnd(work,len1,modulo) # generate random number P0 to use as
                                   #  initial prime factor
     idx2=len1+1 # get index to start of K
     call rmrkln('     Generating K.')
     call genrnd(work(idx2),len2,modulo) # generate random number K
     if ((work(len1) & 1) == 0) # P0 even?
          work(len1)=work(len1)+1 # make P0 odd
     if ((work(lenp) & 1) != 0) # K odd?
          work(lenp)=work(lenp)-1 # make K even
     idxscr=lenp+1 # get index to work scratch area
     lenscr=7*len1+3 # get length of work scratch area
     call remark('     Making P0 Prime.')
     while (! prmtst(work,len1,work(idxscr),lenscr,modulo)) # P0 not prime
          {
          call mpadd(work,len1,two,1,work(idxscr),len1+1,modulo)
                                                       # P0 = P0 + 2
          if (work(idxscr) >= 1) # P0 overflow, i.e. P0 = 10**(LEN1+1) + 1
               work(idxscr+1)=1 # keep P0 between 10**LEN1 & 10**(LEN1+1),
                                #  i.e. P0 --> 10**LEN1 + 1
          do idxo=1,len1 # move new P0 to old P0
               {
               idxn=idxscr+idxo # get index to new P0
               work(idxo)=work(idxn) # new P0 --> old P0
               }
          call rmrkln('.')
          call remark('                   .')
          }
     do idxo=1,len1 # copy P0 to temporary P0 in preparation for division
          {
          idxn=idxscr+idxo # get index to temporary P0
          work(idxn)=work(idxo) # P0 --> temporary P0
          }
     work(idxscr)=0 # make sure numerator has a leading zero
```

```
    idxquo=idxscr+len1+2 # get index to quotient
    lenquo=len1 # get length of quotient
    idxrem=idxquo+lenquo # get index to remainder
    call mpdiv(work(idxscr),len1+1,three,1,work(idxquo),lenquo,work(idxrem),
            1,modulo) # get PO mod 3
    pOmod3=work(idxrem) # save PO mod 3
 do idx0=1,len2 # copy K to temporary K in preparation for division
         {
         idxo=idx2+idx0-1 # get index to K
         idxn=idxscr+idx0 # get index to temporary K
         work(idxn)=work(idxo) # K --> temporary K
         }
    work(idxscr)=0 # make sure numerator has a leading zero
    idxquo=idxscr+len2+2 # get index to quotient
    lenquo=len2 # get length of quotient
    idxrem=idxquo+lenquo # get index to remainder
    call mpdiv(work(idxscr),len2+1,three,1,work(idxquo),lenquo,work(idxrem),
            1,modulo) # get K mod 3
    if (pOmod3 != work(idxrem)) # PO mod 3 != K mod 3?
         {
         # Note: N mod 3 = 0 or 2 or 1 (in that sequence)
         if (pOmod3 == 0) # generate K adjustment for PO mod 3 == 0
              kadj=two*work(idxrem) # adjustment for K mod 3 == 2 or 1
         else if (pOmod3 == 2) # generate K adjustment for PO mod 3 == 2
              kadj=two*(work(idxrem)+one) # adjustment for K mod 3 == 1 or 0
         else # generate K adjustment for PO mod 3 == 1
              kadj=four-work(idxrem) # adjustment for K mod 3 == 0 or 2
         call mpadd(work(idx2),len2,kadj,1,work(idxscr),len2+1,modulo)
                             # force K mod 3 == PO mod 3
         if (work(idxscr) >= 1) # K overflow, i.e. K = 10**(LEN2+1) + 0 or 2
```

(Continued on page 54)

Ad Changed 2/15/84

PROGRAMMERS- COMPARE Products. Get Answers. GUARANTEES

We research and specialize only in software for programmers of micros. If a product is unknown to you and we recommend it, then we take the risk.

PHILOSOPHY: We carry every programmer's product for CPM80, MSDOS and CPM86 plus every key product for APPLE, Commodore 64, ATARI and TRSDOS.

"C" LANGUAGE

		LIST PRICE	OUR PRICE
APPLE:	AZTEC C-Full, ASM	$199	call
8080:	BDS C-Fast, popular	150	125
8080:	AZTEC C-Full	199	call
Z80:	ECOSOFT-Fast, Full	250	225
8086:	**C86**- optimizer, Meg	399	call
8086	Lattice - New1.1& 2.0	600	495
MicroSoft (Lattice)	MSDOS	500	call
Digital Research - Megabyte	8086	350	280
Desmet by CWare-Fast	8086	109	99

BASIC

	ENVIRONMENT		
Active Trace-debug	8080/86	$ 80	73
MBASIC-80 - MicroSoft	8080	375	$255
BASCOM-86 - MicroSoft	8086	395	279
CB-86 - DRI	CPM86/PC	600	449
PBASIC - DRI	8086	150	119
BASIC Dev't System	PCDOS	79	72

EDITORS Programming

BELLESOFT - PASCAL	MSDOS	$ 245	call
C Screen with source	8080/86	NA	60
EDIX	PCDOS	195	149
FINAL WORD	8080/86	300	225
MINCE	CPM, MSDOS	175	149
Micro SPF	PC	250	215
PMATE - powerful	CPM	195	175
	8086	225	195
VEDIT - full, liked	CPM, PCDOS	150	119
	8086	200	159

SERVICES

- **Programmer's Referral List**
- **Compare Products**
- **Help find a Publisher**
- **Evaluation Literature**
- **BULLETIN BOARD** - 7 PM to 7 AM 617-461-0174
- **Dealer's Inquire**
- **Newsletter**
- **Rush Orders**

PASCAL

	ENVIRONMENT	LIST PRICE	OUR PRICE
PASCAL MT+86	CPM86/PC	$400	$289
without SPP	CPM80	350	255
MS PASCAL 86	MSDOS	350	255
SBB PASCAL - great, fast	PCDOS	350	315
PASCAL 64 - nearly full	COM 64	99	89

LANGUAGE LIBRARIES

C to dBASE interface	8080 86	$125	$115
C Tools 1 - String, Screen	PCDOS	NA	115
C Tools 2 - OS Interface	PCDOS	NA	92
C Tools - XOR - Graphics,	Screen	NA	96
FLOAT87 - Lattice, PL1	PCDOS	NA	115
GRAPH GSX-80	CPM80	60	50
HALO	PCDOS	150	125
ISAM Access Manager-86	8086	400	300
PHACT with C	PCDOS	NA	250
FABS	CPM80	150	135
PASCAL TOOLS	PCDOS	NA	115
SCREEN Display Mgr-86	8086	500	375
PANEL-86 - many	PCDOS	NA	315
WINDOWS - AMBER	PCDOS	295	call

FORTRAN

MS FORTRAN-86 - Megabyte	MSDOS	$350	$255
SS FORTRAN-86	CPM-86	425	345
FORTRAN-80 - '66 decent	CPM80	500	350
INTEL FORTRAN-86	IBM PC	NA	1400
DR FORTRAN	COMING		
RM FORTRAN	COMING		

OTHER PRODUCTS

Microshell improve CPM	8080	150	125
IQ LISP - full, 1000K RAM	PCDOS	175	call
PLINK-86 - overlays	8086	350	315
PL/1-86	8086	750	560
Prolog - with samples	PCDOS	NA	125
Lookalike of **Hayes Smart 1200 modem**			**369**
COBOL - Level II	8080/86	1600	1275

NEW FEATURES

For **MICROSOFT C**

Halo - Graphics	$125
Panel - Screen	$315
PHACT - ISAM	$250
C Tools 1 - String	$115
C Tools 2 - OS Interface	$ 92
Others coming	

PRODUCTIVITY USING MSDOS

Assume use of compiler and typical editor. What commercial or public domain products improve productivity? "MSDOS Programming" is a growing document with some answers. Call to request it. **Help improve it.**

Earn $50 credit toward any purchase when we add to "MSDOS Programming" any description, code, or idea received from you.

SUGGESTED PRODUCTS FOR PRODUCTIVITY

- Full use of files across 2.0 directories **AKA ALIAS** $60 Overlays, headers in all directory
- Recover erased or damaged files - use **Norton Utilities** $65 or **POWER** $139 which also includes directory management, memory move, bad sectors file, etc
- Programming tools for filecompare, cross reference, combining/splitting files, linenumber, radix conversion, columnmove, scrollspeed, prettyprint use **Programmer's Helper** $175
- Improve on DEBUG with **Codesmith**-86 $135 many breakpoints/windows, Disasm, save to disk
- Control/run multiple tasks simultaneously. And/ or RAMdisk, better Printspool - **Multijob** $179

Call for a catalog, literature, comparisons, prices. Shipping $2.50 per item purchased. All FORMATS available.

THE PROGRAMMER'S SHOP™

908-D Providence Highway, Dedham, MA 02026.

617-461-0120, Mass: 800-442-8070

VISA 800-421-8006 MASTER CARD

RECENT DISCOVERIES HIGHLIGHTED

C Helper includes source in C for CP/M80, MSDOS for a: DIFF, GREP, Flowcharter, XREF, C Beautifier and others, $115.

Note: all prices subject to change without notice.

Circle no. **43** on reader service card.

Super assemblers plus the world's largest selection of cross assemblers!

Z-80 Macroassembler $49.50

Power for larger programs! This 2500AD macroassembler includes:
- Zilog Z-80 Macroassembler (with the same powerful features as all our assemblers)
- powerful linker that will link up to 400 files
- Intel 8080 to Zilog Z-80 Source Code Converter (to convert all your Intel source to Zilog Syntax in one simple step)
- COM to Hex Converter (to convert your object files to Hex for PROM creation, etc.)
- 52 page User Manual

8086/88 Assembler with Translator $99.50

Available for MSDOS, PCDOS, or CPM/86! This fully relocatable macroassembler will assemble and link code for MSDOS (PCDOS) AND CPM/86 on either a CPM/86 or MSDOS machine. This package also includes:
- An 8080 to 8086 source code translator (no limit on program size to translate)
- A Z-80 to 8086 translator
- 64 page user manual
- 4 linkers included:
  - –MSDOS produces .EXE file
  - –CPM/86 produces .CMD file
  - –Pure object code generation
  - –Object code and address information only

  Linker features:
  - Links up to 128 files
  - Submit mode invocation
  - Code, Data Stack and extra segments
  - Handles complex overlays
- Written in assembly language for fast assemblies.

Z-8000 Cross Development Package $179.50

Instant Z-8000 Software! This package allows development and conversion of software for the Z8001, 8002, 8003 and 8004 based machines on a Z-80, Z-8000 or 8086 machine. This powerful package includes:
- a Z-80 to Z-8000 Assembly Language Source Code Translator
- an 8080 to Z-8000 Source Code Translator
- Z-8000 Macro Cross Assembler
- Linker and Loader
- COM to Hex File Converter
- a 100 page User Manual
- a Zilog Z-8000 Technical Reference Manual

The Translators provide Z-8000 source code from Intel 8080 or Zilog Z-80 source code. This source code expansion is from 2% to 11%. The Translator outputs a worksheet and a Z-8000 source file. The worksheets show each line of 8080 Z-80 code, with notes to help the programmer to optimize performance, and further lower code expansion. It even comments lines it adds! The Z-8000 source code used by these packages are the unique 2500AD syntax using Zilog mnemonics, designed to make the transition from Z-80 code writing to Z-8000 easy.

All 2500 AD Assemblers and Cross Assemblers support the following features:

Relocatable Code—the packages include a versatile Linker that will link up to 128 files together, or just be used for external reference resolution. The Linker allows Submit Mode or Command Invocation.
Large File Handling Capacity—the Assembler will process files as large as the disk storage device. All buffers including the symbol table buffer overflow to disk.
Powerful Macro Section—handles string comparisons during parameter substitutions. Recursion and nesting limited only by the amount of disk storage available.
Conditional Assembly—allows up to 248 levels of nesting.
Assembly Time Calculator—will perform calculations with up to 16 pending operands, using 16 or 32 Bit arithmetic (32 Bit only for 16 Bit products). The algebraic hierarchy may be changed through the use of parentheses.

Include files supported—

Listing Control—allows listing of sections on the program with convenient assembly error detection overrides, along with assembly run time commands that may be used to dynamically change the listing mode during assembly.

Hex File Converter, included—for those who have special requirements, and need to generate object code in this format.

Plain English Error Messages—

System requirements for all programs:
Z-80 CP/M 2.2 System with 64K RAM and at least a 96 column printer is recommended. Or 8086/88 96K CP/M or MSDOS (PCDOS).

Cross Assembler Special Features

Z-8—256 User defined registers names, directive to engage UPC, Z-80 style syntax.

8748—fully Intel source code compatible.

8051—256 User defined register names, bit addressable naming allowed.

6800 Family—absolute or relocatable modes, all addressing modes supported, Motorola syntax compatible.

6502—Standard syntax or Z-80 type syntax supported, all addressing modes supported.

8086 and Z-8000 XASM includes Source Code Translators

	Z-80 CP/M®	ZILOG SYSTEM 8000 UNIX	IBM P.C. 8086/88 MSDOS	IBM P.C. 8086/88 CP/M 86	OLIVETTI M-20 PCOS
8086/88 ASM			$99.50	$99.50	
8086/88 XASM	$179.50	$750.00			$179.50
Z-8000™ ASM		750.00			299.00
Z-8000 XASM	179.50		179.50	179.50	
Z-80 ASM	49.50				
Z-80 XASM		500.00	$99.50	$99.50	$99.50
Z-8 XASM	99.50	500.00	$99.50	$99.50	$99.50
6502 XASM	99.50	500.00	$99.50	$99.50	$99.50
6800,2,8 XASM	99.50	500.00	$99.50	$99.50	$99.50
6801,03 XASM	99.50	500.00	$99.50	$99.50	$99.50
6805 XASM	99.50	500.00	$99.50	$99.50	$99.50
6809 XASM	99.50	500.00	$99.50	$99.50	$99.50
8748 XASM	99.50	500.00	$99.50	$99.50	$99.50
8051 XASM	99.50	500.00	$99.50	$99.50	$99.50
Subtotal	$ ______	$ ______	$ ______	$ ______	$ ______

Name ______________________________

Company ______________________________

Address ______________________________

City ____________ State ________ Zip ________

Phone ____________________ Ext. ________

Make and model of computer system ______________________________

☐ C.O.D. *(2500AD pays C.O.D. charges)*

☐ VISA or MasterCard #, Exp. Date (mo./yr.)

____

TO ORDER. **Simply circle the product or products you want in the price columns above, enter the subtotal at the bottom of that column and add up your total order. Don't forget shipping/handling.**

Check one:

☐ 8″ Single Density
☐ 5¼″ Osborne
☐ IBM P.C.
☐ Cartridge Tape
☐ Apple (Softcard)

Total $ ________

shipping/handling ($6.50 per unit) $ ________

Total Order $ ________

Signature ______________________________

2500AD SOFTWARE INC.

P.O. Box 441410, Aurora, CO 80014, 303-752-4382

Circle no. 1 on reader service card.

RSA (Listing Continued, text begins on page 30)

Listing Seven

```
            work(idxscr+1)=1 # keep K between 10**LEN2 & 10**(LEN2+1),
                             #  i.e. K --> 10**LEN2 + 0 or 2
        do idx0=1,len2 # move new K to old K
            {
            idxo=idx2+idx0-1 # get index to old K
            idxn=idxscr+idx0 # get index to new K
            work(idxo)=work(idxn) # new K --> old K
            }
        }
call mpmult(work,len1,work(idx2),len2,prime,lenp,modulo) # generate K * PO
lenscr=lenp+1 # get length of work scratch area
call mpadd(prime,lenp,one,1,work(idxscr),lenscr,modulo)
            # generate PRIME = K * PO + 1 (10**LENP < PRIME < 10**(LENP+1))
do idxo=1,lenp # move new PRIME to old PRIME
    {
    idxn=idxscr+idxo # get index to new PRIME
    prime(idxo)=work(idxn) # new PRIME --> old PRIME
    }
# Note:  Keeping K mod 3 == PO mod 3 just involves adding 6 to K
#        and (K + 6) * PO + 1 == (K * PO + 1) + (6 * PO)
#                             == PRIME + (6 * PO),
#        i.e. we are done with K !!!
idxscr=idx2 # get index to scratch work area
lenscr=len1+1 # get length of scratch area
call mpmult(work,len1,six,1,work(idxscr),lenscr,modulo) # generate 6 * PO
if (work(idxscr) != 0) # extra digit created?
    len1=len1+1 # get new length of 6 * PO
else # no extra digit created
    idxscr=idxscr+1 # advance index to (6 * PO) to skip over zero digit
    do idxo=1,len1 # move (6 * PO) to old PO
        {
        idxn=idxscr+idxo-1 # get index to (6 * PO)
        work(idxo)=work(idxn) # (6 * PO) --> old PO
        }
    idxscr=len1+1 # get index to scratch work area
    lenscr=lenp+1 # get length of work scratch area for adding (6 * PO)
    lenw=7*lenp+3 # get length of work scratch area for prime testing
    call rmrkln('.')
    call remark('     Generating the prime number.')
    while (! prmtst(prime,lenp,work(idxscr),lenw,modulo)) # PRIME not prime
        {
        call mpadd(prime,lenp,work,len1,work(idxscr),lenscr,modulo)
                # PRIME = PRIME + (6 * PO)
        if (work(idxscr) >= 1) # PRIME overflow,
                               #  i.e. PRIME = 10**(LENP+1) + ?
            work(idxscr+1)=1 # keep PRIME between 10**LENP & 10**(LENP+1),
                             #  i.e. PRIME --> 10**LENP + ?
        do idxo=1,lenp # move new PRIME to old PRIME
            {
            idxn=idxscr+idxo # get index to old PRIME
            prime(idxo)=work(idxn) # new PRIME --> old PRIME
            }
        call rmrkln('.')
        call remark('                                   .')
        }

    #
    # As a precaution at this point, may want to check that (PRIME + 1) does
    #  not consist entirely of rather small prime factors.  If so, should
    #  return to previous WHILE loop to get next PRIME and try again.
    #
    #    There are 168 primes (PRIMEk) less than 1000.  It is sufficient to
    #    show that if factoring PRIME by PRIMEk (k = 1,168) leaves a number
```

```
     #    which is greater than 10**5 then PRIME okay.
     #
     return

end
############################################################################

# GENRND -- Generate a random number of specified length

subroutine genrnd(rnd,lenr,modulo)

     byte rnd(1)
     integer lenr,modulo

     data rndnxt/.5/ # make RAN(*) generate the next random number
                     #   (any positive number will do)

     fmodul=float(modulo) # float byte modulus
     idxr=1 # get index to random number
     repeat # force 1st digit to be non-zero
          irnd=mod(int(fmodul*ran(rndnxt)),modulo) # get a random digit
     until (irnd != 0)
     rnd(idxr)=irnd # load 1st digit of RND
     idxr=idxr+1 # point to next digit of RND
     while (idxr <= lenr) # generate the rest of the random digits
          {
          irnd=mod(int(fmodul*ran(rndnxt)),modulo) # get a random digit
          rnd(idxr)=irnd # load random digit
```

(Continued on next page)

SMALL C FOR IBM-PC

Small-C Compiler Version 2.1 for PC-DOS/MS-DOS
Source Code included for Compiler & Library
New 8086 optimizations
Rich I/O & Standard Library

$40

CBUG SOURCE LEVEL DEBUGGER FOR SMALL C

Break, Trace, and Change variables all on the source level
Source code included

$40

Datalight
11557 8th Ave. N.E.
Seattle, Washington 98125

ASM or MASM is required with compiler.
Include disk size (160k/320k), and DOS version with order.
VISA & MasterCard accepted. Include card no. & expiration date.
Washington state residents include 7.9% sales tax.
IBM-PC & PC-DOS are trademarks of International Business Machines
MS-DOS is a trademark of Microsoft Corporation.

Circle **no. 16** on reader service card.

MU FORTH FOR YOUR IBM PC® ...

Now the advantages of Forth are available for your IBM PC.® Easy to use for both experienced programmers and beginners too, Forth offers the advantages of speed, flexibility, and ease of use.

The MU Forth Package Provides:

- Source and object code user documentation
- Commented source listings in IBM and macro assembler format
- Turtle graphics, sound, and music support
- Keyboard and video support sample programs

Requires:

Your IBM PC, 16K memory, 1 disk drive, PC DOS 1.X, color or monochrome, 40 or 80 column display.

The Price **$75**

Available from: Art Arizpe
MU Software
115 Cannongate III, Nashua, NH 03063
603-880-9416
SEND CHECK OR MONEY ORDER

Circle **no. 38** on reader service card.

RSA (Listing Continued, text begins on page 30)

Listing Seven

```
            idxr=idxr+1 # advance pointer to next digit
            }
      return

end
```

End Listing Seven

Listing Eight

```
# PROGRAM NAME:  EDCRYPT.RAT
# PURPOSE:  This routine uses the RSA (Rivest-Shamir-Adleman) keys generated
#           by GENPKEYS.RAT to Encrypt/Decrypt a message.
#           It uses the method described by D.E. Knuth, The Art of Computer
#           Programming, V. 2 (Semi-Numerical Algorithms), 2nd Ed., (Addison-
#           Wesley, Reading, MA), pp. 386-389.
#
# LANGUAGE:  RATFOR
# AUTHOR:  CEB
# USAGE:  ******** (main program)
#
#
#
# ARRAYS USED:  N(KEYLENGTH),D(KEYLENGTH),WORK(WORKLENGTH)
# EXTERNALS:  RPEXP
# UPDATE HISTORY:  INITIAL RELEASE -- 02/14/83 CEB
#

define(KEYLENGTH,90) # length of key for characters
define(WORKLENGTH,450) # length of WORK buffer for characters
                       #  (5 * KEYLENGTH)
define(BYTEMODULUS,128) # byte modulus for characters (1 character/byte)

define(LUNIN,6) # Input Logical Unit
define(LUNOUT,7) # Output Logical Unit
define(BLOCKS,8) # Number of 128 byte blocks to use for disk I/O

define(EOFCHAR,26) # text file EOF character (^Z)

program rsa

      character c,c0,getc,getch,oper,type,extn(3),linein(KEYLENGTH),
                lineot(KEYLENGTH)
      byte n(KEYLENGTH),d(KEYLENGTH),work(WORKLENGTH),three,msb
      integer openr,equal,getl
      double precision filnam # allow entry of up to 8 characters

      equivalence (filnam,work(1))

      data extn/BIGT,BIGX,BIGT/ # default file extension
      data modulo/BYTEMODULUS/ # get byte modulus for arithmetic
      data three/3/
      call rmrkln('.') # show options
      call rmrkln('Select operation:.')
      call rmrkln('      <E>ncryption.')
      call rmrkln('      <D>ecryption.')
      call rmrkln('.')
      repeat # get operation to perform
            {
```

```
            call remark('Operation -- .')
            oper=getc(c) & 95 # get selection and convert to upper case
            call rmrkln('.')
            }
      until ((oper == BIGE) | (oper == BIGD)) # valid entry
      call open(LUNIN,'keys    dat',0) # open key file on current drive
      read(LUNIN,100) lenn # get the length of N
100   format(i3)
      read(LUNIN,200) (n(idx), idx=1,lenn) # get N
200   format(20(i3,1x))
      if (oper == BIGD) # decryption selected?
            {
            read(LUNIN,100) lend # get length of D
            read(LUNIN,200) (d(idx), idx=1,lend) # get D (cube root of text)
            }
      else # encryption selected
            {
            lend=1 # only one digit
            d(lend)=three # set-up to cube text
            }
      endfile LUNIN # close key file
      repeat # get name of text file (.TXT extension assumed)
            {
            call rmrkln('.')
            if (oper == BIGD) # decryption selected?
                  call remark('Encrypted.')
            else # encryption selected
                  call remark('Clear.')
            call remark(' Text Source File Name (XXXXXXXX) -- .')
            read(CONSOLE,300) filnam # WORK(1 to 8)
300         format(a8)
            do idx=1,8 # convert lower case to upper case
                  if (type(work(idx)) == LETTER) # is character a letter?
                        work(idx)=work(idx) & 95 # to upper case
            work(9)=EOS # terminate file name
            }
      until (openr(LUNIN,READONLY,BLOCKS,work,extn) == YES) # successful open
      call scopy(work,1,work,10) # save source file name
      repeat # get name of text file (.TXT extension assumed)
            {
            call rmrkln('.')
            if (oper == BIGD) # decryption selected?
                  call remark('Clear.')
            else # encryption selected
                  call remark('Encrypted.')
            call remark(' Text Destination File Name (XXXXXXXX) -- .')
            read(CONSOLE,300) filnam # WORK(1 to 8)
      do idx=1,8 # convert lower case to upper case
            if (type(work(idx)) == LETTER) # is character a letter?
                  work(idx)=work(idx) & 95 # to upper case
      work(9)=EOS # terminate file name
      if (equal(work,work(10)) == YES) # Source & Destination names same?
            {
            call rmrkln('.')
            call remark('Source and Destination names cannot be the .')
            call rmrkln('same !!!.')
            }
      }
until (openr(LUNOUT,WRITEONLY,BLOCKS,work,extn) == YES) # successful open
for (c0=0; c0 != EOF; ) # convert text
      {
      if (oper == BIGE) # encryption?
            {
            linein(1)=0 # make sure LINEIN mod N = LINEIN
            len=2 # point to next character
            }
      else # decryption
            len=1 # initialize line length counter
```

(Continued on next page)

Listing Eight

```
          repeat # get a line of text (encrypted or decrypted)
               {
               linein(len)=getch(c0,LUNIN) & 127 # get a character/strip MSBit
                       # (no effect on Clear Text but MSBit set by encryption)
               len=len+1 # point to next character position
               if ((oper == BIGE) & (c0 == EOF)) # encryption & EOF?
                    {
                    linein(len-1)=EOFCHAR # overwrite EOF with EOFCHAR
                    while (len <= lenn) # clear text line
                         {
                         repeat
                              num=int(32.0*ran(.5)) # get a number (0 - 31)
                         until (num != EOFCHAR) # not text EOF character
                         linein(len)=num # pad line with a random character
                                        #  (could have previously seeded
                                        #  random number generator !!!)
                         len=len+1 # advance line length counter
                         }
                    }
               }
           until ((c0 == EOF) | (len > lenn)) # ready to encrypt/decrypt
           if ((oper == BIGD) & (c0 == EOF)) # decryption & EOF?
                break # encryption does not leave characters with EOF, so done
           lenin=len-1 # point to last actual character
           lenout=lenn # get length of converted line
           lenw=4*lenn+3 # get length of work area
           call rpexp(linein,lenin,d,lend,lineot,lenout,n,lenn,work,lenw,modulo)
                          # encrypt/decrypt line
      if (oper == BIGD) # decryption selected?
           {
           len0=2 # skip over leading zero from encryption
           msb=0 # MSBit for decrypted text is zero
           if (c0 == EOF) # end of file found on input?
                {
                while ((lineot(lenout) != EOFCHAR) & (lenout >= len0))
                                                  # strip pad characters
                     lenout=lenout-1 # backup end of line pointer
                lenout=lenout-1 # backup over EOFCHAR
                }
           }
      else # encryption
           {
           len0=1 # use all characters
           msb=128 # MSBit for encrypted text is one for Control Chars.
           }
      if (lenout >= len0) # at least one character to output?
           {
           do idx=len0,lenout # save encrypted/decrypted text
                {
                c=lineot(idx) # get character to save
                if (c < BLANK) # Control Character (only used during
                               #  encryption)?
                     c=c | msb # move Control Characters above printable
                               #  ASCII range, especially EOFCHAR !!!
                call putch(c,LUNOUT)
                }
           }
      }
 call closer(LUNIN) # close files
 call closer(LUNOUT)
end
```

End Listing Eight

Listing Nine

```
#     Random Number Generator for use with APULIB
#
#           Modeled after W.J. Cody & W. Waite, "Software Manual for the
#                 Elementary Functions"
#

real function ran(x)

      real x,y,const,zero

      data y/100001.0/,const/2796203.0/,zero/0.0/

      if (x == zero)
            ran=y
      else
            {
            if (x < zero)
                  y=-x*const
            y=amod(125.0*y,const)
            ran=y/const
            if (y == zero)
                  y=100001.0
            }
      return

end
```

End Listings

THE FULL-FEATURED KEYBOARD EXPANDER
for all 8080-8085-Z80 computers using CP/M 2.2

MagiKey™ will redefine your keys as character strings ... and transform single keystrokes into commands for programs, words for word processors, data for data bases, and messages for modems. Without hardware or system modifications.

MagiKey™ has more advanced features than any other CP/M keyboard expander. For example:

★ Redefine a key to send the string:
"Run WordStar and edit form letter number 17"
Hit the key. YOU will see the string, but "WS FRMLTR17" will be sent to CP/M. **MagiKey™**'s CONSOLE REDIRECTION can display messages which are invisible to programs and CP/M ... very useful for recalling key assignments, custom operator prompts, and making CP/M friendlier.

★ Redefine a key to run several programs in sequence. Each program can wait for keyboard input or receive pre-defined commands and data. **MagiKey™**'s built-in BATCH PROCESSING doesn't use CP/M's SUBMIT, and handles programs that CP/M's XSUB can't.

★ Redefine a key to display the prompt:
"Execute SuperCalc using the spreadsheet file:"
Press the key to display the prompt. Type the file name, hit RETURN, and you're into SuperCalc. When done, a single keystroke saves your updated spreadsheet. You don't have to remember or retype its name. **MagiKey™**'s RECURSIVE key redefinition mode automatically does it for you.

WE INVITE COMPARISON

$100
8" SSSD, most 5¼" formats
add 6% tax in CA
check, VISA, M/C

CP/M (tm) Digital Research
SuperCalc (tm) Sorcim
WordStar (tm) Micropro

PRO microSystems
16609 Sagewood Lane
Poway, California 92064
(619) 693-1022

Circle **no. 44** on reader service card.

COMPUTER RESOURCES of WAIMEA
Menu Driven Programs running under SPELLBINDER

DearJohn will free your operators from entry errors. Completely menu driven Mailing List Management Programs with thoughtfully formatted screens to ease the entry of addresses and names. Integrity of the data file created is automatically verified. No more printing 'bad' mailing labels with misplaced data; AND each address record can be individually classified to allow inclusion (or exclusion) when printing labels or form letters.
(10 Programs) $69.50

SBMenu SBHelp make your version of Spellbinder menu driven. No more fumbling through manuals for obscure or little-used commands. The complete instruction set can be displayed at any time without disturbing the text on the screen. These two programs put Spellbinder into the world class of word processors, allowing the user to easily use its truly outstanding features.
(With two help files) $55.00

AutoSave automatically saves the text on the screen to a previously named file. Prevents the inadvertent but all so common save to a wrong name. Works with source code and program code as well as with text files. All instructions on disk.
(3 Programs) $37.50

Send Check, Money Order, Visa, Master Charge:
COMPUTER RESOURCES OF WAIMEA
P.O. Box 1206
Kamuela, Hawaii 96743
Phone: (808) 885-7905

Circle **no. 11** on reader service card.

BASICFMT for TRS-80

[A BASIC Formatter]

OK, programmer! It's time to play "Find the Variable"! The object of the game is to determine what the variables are in line 10 of my program. The winner gets a Rubik's Cube!

10 IFBASICANDCOMPRESSEDTHEN SANITY=CRAZYANDYESELSEIFNOT BASICANDCOMPRESSEDTHENSANIT Y=CRAZYANDNO

by Davy Crockett

Davy Crockett, 5807 Cherrywood Ln., Apt. 104, Greenbelt, MD 20770

The Game

Many programmers like to play games with their programs. I do not mean after you type RUN. These games are played when RUN would only elicit a READY prompt from the BASIC interpreter. They choose variable names like DO. That way, when you use it in an IF statement it can look like this:

100 IF DOOR KNOB THEN 300

Of course, judicious use of spacing could clarify the issue, but that is contrary to the rules of the game.

Now I know what you are thinking: "This guy is leading up to yet another program to format a BASIC ASCII file." You are correct on one count: this is a formatting program. However, it is not just another program. BASICFMT reads a BASIC program in *compressed* format from disk and prints a readable listing. In order to use BASICFMT, all you need is a BASIC program stored on disk without using the ",A" option. No longer is it necessary to load the program and then save it again with the ASCII option.

The Game Pieces

The first byte of these BASIC files is always 0FFH. This byte tells BASIC that it is all right to continue loading this file as a BASIC program. If this byte is not 0FFH then you get the old "Direct Statement in File" error from BASIC. Following this byte, each line is stored in

```
10 '
        ***********************  Version 1.0
        *****  DEMO/BAS   *****  29 October 1982
        ***********************  Davy * Crockett

1510 '
        Davy * Crockett, B.S., Esquire
        5807 Cherrywood Lane, Apt. 104
        Greenbelt, Maryland      20770

3010 '
        Program to demonstrate BASICFMT

4510 CLS:CLEAR1000:DEFINTA-Z:DIMNM$(35),AD$(35),TL$(2):GOSUB13510:ONER
RORGOTO12010
6010 DEFFNLN$=LEFT$(NM$(I),INSTR(1,NM$(I),",")-1)
7510 GOTO10510:REM - remark statement
9010 CLS:PRINTTL$(1);:PRINTTL$(2):RETURN
10510 GOSUB9010:I=35:PRINT"Please enter your name (last, first)":INPUT
NM$(I),NM$(I-1):NM$(I)=NM$(I)+", "+NM$(I-1):LN$=FNLN$:PRINT:PRINT"Hell
o, ";LN$:FORK=0TO200:NEXT:GOTO10510
12010 PRINT"ERL =";ERL;"   ERR =";ERR:ONERRORGOTO0
13510 FORK=1TO2:FORM=1TO21:READY:TL$(K)=TL$(K)+CHR$(Y):NEXT:NEXT:RETUR
N
15010 DATA191,131,171,148,32,191,179,179,131,32,191,140,176,140,191,32
,191,131,131,191,10,143,140,142,129,32,143,140,140,140,32,143,32,32,32
,143,32,143,140,140,143,10
```

Figure 1.

BASIC'S LLIST command

a specific format.

The first two bytes of each line contain an address which points to the line number of the following line. The second pair of bytes is the line number in binary format. After this is the actual statement with all of the reserved words compressed into a single byte "token." A binary zero (00H) is the end-of-line marker. For example, the preceding line 100 would, in compressed format on disk, appear as in Figure 1 (page 60). The end of the program is indicated by a next-line pointer consisting of two binary zeros.

Playing The Game

The BASIC tokens are all hex codes which cannot be generated by simply pressing keys on the keyboard. This makes them easy to identify in a section of compressed BASIC code. The character strings associated with each token are stored in the ROM at location 1650H for both the Model I and the Model III. Once a token is found, it is a simple matter to extract the character string associated with that token from the table stored in the ROM and separate it by blanks before printing.

The first thing I noticed is that BASIC does not always compress a program the way I expected it to, particularly the "remark" commands. The characters REM get translated to a token (93H) as you would expect. However, when I used an apostrophe to indicate a remark, three bytes were inserted instead of a single byte token. The first byte was a colon, the second byte was the remark token, and the third byte was the hex value 0FBH. The 0FBH apparently tells BASIC to back up to the preceding colon and replace it with an apostrophe when listing the program.

The ELSE command contained in an IF statement is another reserved word that is not treated as you would expect. The ELSE command is always converted to a two-byte sequence with a colon preceding the ELSE token.

Winning At The Game

BASICFMT can help beat these gamesters at their own game by separating the variables and the command words by blanks. The program opens by asking for the filespec of the BASIC file to be formatted. It will then open the file and check to make sure that it is a BASIC program in compressed format. If the file is not in the expected format, an error message is printed and you are returned to the DOS.

When the file checks out, you will be asked for printer specifications. These include: maximum line length, number of lines to print on a page, and number of lines to skip between pages. Printing begins immediately upon entry of the last printer specification. If the printer is not ready, the program will print a mes-

```
   10 '
            ***********************  Version 1.0
            *****  DEMO/BAS   *****  29 October 1982
            ***********************  Davy * Crockett

 1510 '
            Davy * Crockett, B.S., Esquire
            5807 Cherrywood Lane, Apt. 104
            Greenbelt, Maryland      20770

 3010 '
            Program to demonstrate BASICFMT

 4510 CLS : CLEAR 1000 : DEFINT A - Z : DIM NM$(35), AD$(35), TL$(2)
      : GOSUB 13510 : ON ERROR GOTO 12010
 6010 DEF FN LN$ = LEFT$ (NM$(I), INSTR (1, NM$(I), ",") - 1)
 7510 GOTO 10510 : REM - remark statement
 9010 CLS : PRINT TL$(1); : PRINT TL$(2) : RETURN
10510 GOSUB 9010 : I = 35 : PRINT "Please enter your name (last,
      first)" : INPUT NM$(I), NM$(I - 1) : NM$(I) = NM$(I) + ", " +
      NM$(I - 1) : LN$ = FN LN$ : PRINT : PRINT "Hello, "; LN$ : FOR
      K = 0 TO 200 : NEXT : GOTO 10510
12010 PRINT "ERL ="; ERL ; "   ERR ="; ERR : ON ERROR GOTO 0
13510 FOR K = 1 TO 2 : FOR M = 1 TO 21 : READ Y : TL$(K) = TL$(K) +
      CHR$ (Y) : NEXT : NEXT : RETURN
15010 DATA 191, 131, 171, 148, 32, 191, 179, 179, 131, 32, 191, 140,
      176, 140, 191, 32, 191, 131, 131, 191, 10, 143, 140, 142, 129,
      32, 143, 140, 140, 140, 32, 143, 32, 32, 32, 143, 32, 143, 140,
      140, 143, 10
```

Figure 2.

BASICFMT output

sage for you to ready the printer and then wait for you to press ENTER.

The target program is read by a routine (GETBYT) which returns a single byte from the disk buffer each time it is called. Some programming languages, notably PL/I, call this technique "character stream" input. One line at a time is read this way, decompressed, and stored in a primary buffer. Extra blanks separating the command words, colons, semi colons, and commas are inserted in the statement at this point. All remark statements and things enclosed within quotation marks are not changed in any way.

Phase two consists of moving the contents of the primary buffer to a secondary buffer. While this is being done, the line breaks are inserted in the statement. Each line is broken at a blank if possible. This would not be possible if there were a remark or quoted string which had no blanks and completely filled the last 10 characters on the line. In this case, the line is broken at the maximum line length specified. Subsequent lines of the same statement are indented past the line number to make the line number easy to read.

Finally, each line in the secondary buffer is printed. The page length is checked before each line to make sure it will not exceed the maximum number of lines specified as page length. Processing continues with the next line until the end of the file is reached. You are then returned to the DOS READY prompt.

Game Strategy

This program will run under any DOS which uses the standard entry points to open an existing file (4424H), read a sector (4436H), and close a file (4428H). It will also run on either the Model I or the Model III. This is due to several factors. First, the BASIC token table is located at the same address in ROM for both models. Second, I used the assembler instruction LD A,(37E8H) to check the printer status instead of IN A,(0FBH), which only works on the Model III. Third, many of the DOS/ROM routines are incorporated in the program so that you do not have to worry about them when switching from one DOS to another. These include keyboard input, screen and printer output, ASCII characters to binary value, binary value to ASCII decimal characters, and even a routine to simulate the single-byte read from disk available with NEWDOS/80.

Some of you may wish to add a routine that prompts for a title line to be printed at the top of each page. I did not feel that this was necessary because I always include an identification section for each program at the beginning. You could also modify the PAGER routine to print a page number when it ejects a page. If your printer requires something separating carriage returns or line feed characters, the LFEED routine will have to be modified to transmit the required codes.

Before I forget, this statement may be easier to read:

```
10  IF BASIC AND COMPRESSED
    THEN SANITY = CRAZY AND YES
    ELSE IF NOT BASIC AND
    COMPRESSED THEN SANITY =
    CRAZY AND NO
```

or in other words:

```
10  IF BA AND CO THEN SA = CR AND
    YE ELSE IF NOT BA AND CO
    THEN SA = CR AND NO
```

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 195.**

Inter-Office Correspondence Made Simple

SEATTLE OFFICE

```
Memo editor: Enter mail                               Mode: Add
Date: 11/30/83
                                          Special delivery: No
From mailbox: 709 - Jack Spears             Return receipt: Yes
To mailbox:   101 - Will Pierce                  Copies to: None
Subject: Electronic Mail Manager            Attached files: No
-----------------------------------------------------------------
Will,

     Since the installation of the "EMM" system, our
inter-office correspondance has become effective and
more productive.  The system is a real "bargan" at
even 10 times the price!  See you next week.
```

LYME STREET OFFICE

```
Memo editor: Enter mail                               Mode: Add
Date: 11/30/83
                                          Special delivery: Yes
From mailbox: 101 - Will Pierce             Return receipt: Yes
To mailbox:   102 - John Fail                    Copies to: 5 mailboxes
Subject: Board Meeting tomorrow             Attached files: Yes
-----------------------------------------------------------------
Hi John,

        The board meeting will be held tomorrow morning at 10
 A.M.  This Electronic Mail Manager sure makes communications
 between our offices easy and fast. The new budget forcast is
 attached in xyCALC format.  Let me know if you approve.
```

EMM™ PLUS COMMX™

LA OFFICE

```
Memo editor: Enter mail                               Mode: Add
Date: 11/30/83
                                          Special delivery: Yes
From mailbox: 102 - John Fail               Return receipt: Yes
To mailbox:   101 - Will Pierce                  Copies to: None
Subject: Board Meeting tomorrow             Attached files: Yes
-----------------------------------------------------------------
Will,

     The budget forcast looks good except that I have
revised the year end projections to include  new market
trend figures.  I have returned the new setup in xyCALC.
See you at tomorrows meeting!
```

SOUTH STREET OFFICE

```
Memo editor: Enter mail                               Mode: Add
Date: 11/30/83
                                          Special delivery: No
From mailbox: 908 - Lara Wilks              Return receipt: Yes
To mailbox:   101 - Will Pierce                  Copies to: All
Subject: New Product Release                Attached files: No
-----------------------------------------------------------------
Will,

     We have completed the integration testing of the
fortran version of COMMX on the DEC 10 with links to
all the micros.  The error free file transfers work
like a dream!  The COMMX protocol does it again!
```

Effective Organizational Communications can give YOU the Competitive Edge. Get EMM™ Electronic Mail Manager and COMMX™ COMMunications eXchange software on YOUR TEAM today! Set up this LAM/NET™ Logical Access Modem Network to meet your Inter-Office our Inter-Office requirements on CP/M80, CP/M86 and MSDOS. dBase II™ Runtime™ demo disk $25 ppd.

Trademarks: dBase II & RunTime (tm) Ashton Tate, CP/M80 & CP/M86 (tm) Digital Research, MSDOS (tm) Microsoft, EMM & COMMX & LAM/NET (tm) Hawkeye Grafix.

HAWKEYE GRAFIX
23914 Mobile Street, Canoga Park, CA 91306 • (213) 348-7909 or 634-0733 Modem (300 B): 634-0441

Circle **no. 26** on reader service card.

BASICFMT (Text begins on page 60)

```
00010 ;=========================================================*
00020 ;                                                         *
00030 ;        *********************  Version 3.4               *
00040 ;        *****  BASICFMT  *****  01 November 1983          *
00050 ;        *********************  Davy * Crockett           *
00060 ;                                                         *
00070 ;        Davy * Crockett, B.S., Esquire                   *
00080 ;        5807 Cherrywood Lane, Apt. 104                   *
00090 ;        Greenbelt, Maryland      20770                   *
00100 ;                                                         *
00110 ;        This program will print a formatted listing      *
00120 ;        of a basic program from disk.  The program       *
00130 ;        must not, I repeat "NOT", be stored in ascii     *
00140 ;        format.  BASICFMT prompts for basic filespec,    *
00150 ;        printer line length, lines per page, and         *
00160 ;        spacing between pages for the output.            *
00170 ;                                                         *
00180 ;=========================================================*
00190 ;
00200 ;>>==>  System equates section
00210 ;
00220 TOKTAB  EQU     1650H   ;basic's token table address
00230 LPPORT  EQU     37E8H   ;line printer status port
00240 CTLKEY  EQU     3840H   ;control key address
00250 OPEN    EQU     4424H   ;open existing disk file
00260 READ    EQU     4436H   ;read a record from disk
00270 CLOSE   EQU     4428H   ;close a disk file
00280 DOSERR  EQU     4409H   ;display DOS error
00290 DOS     EQU     402DH   ;return to operating system
00300 ;
00310         ORG     5800H
00320 ;
00330 BASFMT  LD      SP,BASFMT           ;initialize stack
00340 GETFS   LD      DE,FSMSG            ;point to prompt
00350         LD      HL,DCB              ;point to FCB
00360         LD      B,32                ;maximum chars
00370         CALL    INPUTP              ;get filespec
00380         CP      1                   ;<BREAK>?
00390         JP      Z,DOS               ;yes, exit
00400         LD      DE,DCB              ;point to FCB
00410         LD      HL,BUFFER           ;point to buffer
00420         LD      B,0H                ;LRECL=256
00430         CALL    OPEN                ;open the file
00440         JP      NZ,DOSERR           ;error, exit
00450         CALL    GETBYT              ;get a byte
00460         CP      0FFH                ;BASIC file?
00470         JR      Z,GETLL             ;yes, continue
00480         LD      HL,FYLMSG           ;no, point to error msg
00490         CALL    PRINT               ; and display message
00500         JR      GETFS               ;return to get filespec
00510 ;
00520 ;>>==>  Get printer characteristics from operator
00530 ;
00540 GETLL   LD      DE,LLMSG            ;point to prompt
00550         CALL    GETVAL              ;get line length
00560         JR      Z,GETPL             ;got something?
00570         SUB     6                   ;adjust line length
00580         LD      (LL),A              ; and store it
00590 GETPL   LD      DE,PLMSG            ;point to prompt
00600         CALL    GETVAL              ;get page length
```

(Continued on next page)

```
00610          JR      Z,GETPS         ;got something?
00620          LD      (PL),A          ;store page length
00630 GETPS    LD      DE,PSMSG        ;point to prompt
00640          CALL    GETVAL          ;get page spacing
00650          JP      Z,BEGIN         ;got something?
00660          LD      (PS),A          ;yes, store it
00670          JP      BEGIN           ; and start
00680 ;
00690 ;>>==>   Obtain a decimal value from the keyboard
00700 ;       and convert it to a binary value.
00710 ;
00720 GETVAL   LD      HL,LINE         ;point to buffer
00730          LD      B,4             ;maximum chars
00740          CALL    INPUTP          ;get characters
00750          CP      1H              ;<BREAK>?
00760          JR      Z,EXIT          ;yes, exit
00770          CALL    DECBIN          ;convert to binary
00780          LD      A,E             ;get binary value
00790          OR      A               ;set/reset Z flag
00800          RET
00810 ;
00820 DONE     LD      A,0CH           ;form feed
00830          CALL    LPBYTE          ;eject page
00840 EXIT     LD      DE,DCB          ;point to FCB
00850          CALL    CLOSE           ;close the file
00860          JP      DOS             ;exit, stage left
00870 ;
00880 ;>>==>   Prompt and information messages section
00890 ;
00900 FSMSG    DEFW    1F1CH           ;home & cls
00910          DEFM    '=*= Basicfmt =*= by Davy * Crockett'
00920          DEFW    0D0DH           ;couple a <CR>'s
00930          DEFM    'Enter source filespec'
00940          NOP
00950 LLMSG    DEFB    0DH
00960          DEFM    '* File successfully opened'
00970          DEFW    0D0DH
00980          DEFM    'Enter line length (<ENTER>=85)'
00990          NOP
01000 PLMSG    DEFB    0DH
01010          DEFM    'Enter page length (<ENTER>=60)'
01020          NOP
01030 PSMSG    DEFB    0DH
01040          DEFM    'Enter page spacing (<ENTER>=6)'
01050          NOP
01060 FYLMSG   DEFB    0DH
01070          DEFM    '>>==> File not compressed BASIC'
01080          NOP
01090 ;
01100 ;        *********************************
01110 ;        * Main program control section. *
01120 ;        *********************************
01130 ;
01140 BEGIN    CALL    LPSTAT          ;check printer status
01150 NXTLYN   CALL    GETLYN          ;buffer a statement
01160          CALL    PUTLYN          ;print a statement
01170          JR      NXTLYN          ;go get another
01180 ;
01190 ;>>==>   Store a statement number in the primary buffer
01200 ;
```

```
GETLYN  CALL    GETBYT          ;get a byte
        LD      L,A             ; and store in (L)
        CALL    GETBYT          ;get another byte
        LD      H,A             ; and store in (H)
        LD      DE,0            ;test value
        CALL    CPHLDE          ;zero address?
        JP      Z,DONE          ;yes, quit
        LD      DE,LINE         ;point to buffer
        CALL    GETBYT          ;LSB line number
        LD      L,A             ; and store in (L)
        CALL    GETBYT          ;MSB line number
        LD      H,A             ; and store in (H)
        CALL    BINDEC          ;convert to decimal
        LD      DE,LINE+5       ;end of line #
        LD      A,' '           ;get a blank
        LD      (DE),A          ; and store after #
        XOR     A               ;set the quote
        LD      (QUOTE),A       ; indicator
        LD      (REMARK),A      ; and rem indicator
;
;>>==>  Store the rest of the statement in the buffer
;
BUILD   CALL    GETBYT          ;get a byte
        OR      A               ;end of the line?
        RET     Z               ;yes, return
        CP      80H             ;token?
        JR      NC,TOKEN        ;yes, jump
CHK22   CP      '"'             ;quote?
```

(Continued on next page)

GGM – FORTH™ has HELP*
for Z80[1] using CP/M[2]

GGM–FORTH, a complete software system for real-time measurement and control, runs on any Z80 computer under CP/M using an extended fig-FORTH vocabulary.

GGM–FORTH features:

- Open multiple CP/M files, in any combination of direct-access and sequential-access, fully compatible with all CP/M utilities
- Char. in/out uses CP/M console, lister, file, or port
- On-line HELP* provides instant access to definitions in the run-time GGM–FORTH dictionary
- HELP* file is easily extended to include user definitions using HELP* utility
- HELP* is available during full-screen editing

Complete system and manuals	$150.
Manuals only:	$ 20.
Introductory System:	$ 35.

GGM SYSTEMS, INC. (617) 662-0550
135 Summer Ave., Reading, MA 01867

[1] Z80 is a trademark of Zilog, Inc.
[2] CP/M is a trademark of Digital Research, Inc.

Circle **no. 22** on reader service card.

At Last! bds C . . .

Ver.1.5

Including a new dynamic debugger
Still the choice of professionals

- Compiler option to generate special symbol table for new dynamic debugger by David Kirkland. (With the debugger, the distribution package now requires two disks.)
- Takes full advantage of CP/M® 2.x, including random-record read, seek relative to file end, user number prefixes, and better error reporting.
- Clink option to suppress warm-boot
- New library file search capabilities
- New, fully-indexed 180 page manual
- ® CP/M is a trademark of Digital Research, Inc.

V 1.5 . . . . . $120.00
V 1.46 . . . . $115.00
(needs only 1.4 CP/M)

Other C compilers and C related products available . . . Call!

TERMS: CHECK, MONEY ORDER, C.O.D., CHARGE CARD
HOURS: 9 am—5 pm
Monday —Friday
(316) 431-0018

IT'S HERE!
MONEY MATH
- Uses BCD internal representation.
- You choose from two types of rounding.
- Configurable exception handling
- Distributed with 12 digits precision. Easily configured for more or less
- Excess 64 exponents

SOURCE INCLUDED $50.00

DMS
Dedicated Micro Systems, Inc.
P.O. Box 481, Chanute, Kansas 66720

include $2.50 for postage and handling

Circle **no. 18** on reader service card.

WHY FORTH ?

- Genuinely Interactive (BASIC is much less interactive)
- Encourages Modular Programs (inefficiency and cluttered syntax hamper effective modularization in compiled languages)
- Fast Execution (not even C is faster)
- Amazingly Compact Code
- Fast Program Development
- Easy Peripherals Interfacing

HS/FORTH

- Fully Optimized for IBM-PC IBM-XT IBM-JR and all PCDOS compatibles
- Forth-79 and Forth-83 Modes
- Full Support for DOS Files, Standard Screens and Random Access DOS Screen Files
- Full Use of 8088 Instructions (not limited 8080 conversion subset of transported versions)
- Separate Segments for Code, Stack, Vocabularies, and Definition Lists - multiple sets possible
- Segment Management Support
- Data Anywhere in Full Megabyte
- Coprocessor Support
- Multi-task, Multi-user Compatible
- Automatic Optimizer (no assembler knowledge needed)
- Full Assembler (interactive, easy to use & learn)
- Goal Oriented Documentation
- Free Updates and News until June 1985
- BYTE Sieve Benchmark jan83
  HS/FORTH 47 sec BASIC 2000 sec
  w/AUTOOPT 9 sec Assembler 5 sec
  other Forths (mostly 64k) 70-140 sec

PS You don't have to understand this ad to love programming in HS/FORTH!

HS/FORTH with AUTO-OPT & MICRO-ASM $220.

HARVARD SOFTWORKS

PO BOX 339
HARVARD, MA 01451
(617) 456-3021

Circle **no. 25** on reader service card.

BASICFMT (Listing Continued, text begins on page 60)

```
01490          JR     NZ,CHK10        ;no, continue
01500          PUSH   AF              ;save (AF)
01510          LD     A,(QUOTE)       ;get quote value
01520          XOR    1               ;switch it
01530          LD     (QUOTE),A       ; and put it back
01540          POP    AF              ;restore (AF)
01550          JR     CONT            ; and continue
01560 CHK10    CP     0AH             ;line feed?
01570          JR     NZ,VALID        ;no, check validity
01580          LD     A,0DH           ;replace w/ <CR>
01590          JR     CONT            ; and continue
01600 VALID    CP     ' '             ;valid char?
01610          JR     C,BLANK         ;no, zap it
01620          CP     ':'             ;colon character?
01630          CALL   Z,SPACE         ;yes, add space
01640 CONT     INC    DE              ;bump pointer
01650          LD     (DE),A          ;store character
01660          CP     ','             ;comma?
01670          JR     Z,CONT1         ;yes, insert blank
01680          CP     ';'             ;semi-colon?
01690          JR     Z,CONT1         ;yes, insert blank
01700          CP     ':'             ;colon character?
01710 CONT1    CALL   Z,SPACE         ;yes, add a space
01720 CONT2    JR     BUILD           ;go for next
01730 BLANK    LD     A,' '           ;zap the char
01740          JR     CONT            ; and continue
01750 ;
01760 ;>>==>  Obtain characters represented by basic token
01770 ;       from the token table in the basic ROM.
01780 ;
01790 TOKEN    CP     0FCH            ;garbage char?
01800          JR     NC,BLANK        ;yes, zap it
01810          PUSH   AF              ;save token
01820 CHK0FB   CP     0FBH            ;zap REM token?
01830          JR     NZ,CHK95        ;no, continue
01840          DEC    DE              ;backup
01850          DEC    DE              ; to the
01860          DEC    DE              ;  prior
01870          DEC    DE              ;    colon
01880          LD     A,27H           ;change colon
01890          LD     (DE),A          ; to quote (')
01900          POP    AF              ;clear the stack
01910          JR     BUILD           ;go get next character
01920 CHK95    CP     95H             ;ELSE token?
01930          JR     NZ,CHK93        ;no, continue
01940          DEC    DE              ;yes, delete the prior
01950          DEC    DE              ;  space and colon
01960          JR     OFFSET          ;    and continue
01970 CHK93    CP     93H             ;REM token?
01980          JR     NZ,OFFSET       ;no, compute offset
01990          LD     (REMARK),A      ;set REM indicator
02000 OFFSET   SUB    7FH             ;get position in table
02010          LD     B,A             ; and put in (B)
02020          LD     HL,TOKTAB       ;point to token table
02030 TOKEN1   LD     A,(HL)          ;get table character
02040          INC    HL              ;increment the pointer
02050          CP     7FH             ;first character of a token?
02060          JR     C,TOKEN1        ;no, get next character
02070          DJNZ   TOKEN1          ;yes, loop until token
02080          SUB    80H             ;got it, adjust first char
```

```
        CALL    SPACE           ;insert space (maybe)
TOKEN2  INC     DE              ;increment buffer pointer
        LD      (DE),A          ;and store in buffer
        LD      A,(HL)          ;get next character
        INC     HL              ;increment token pointer
        CP      7FH             ;end of token?
        JR      C,TOKEN2        ;no, continue moving token
        POP     AF              ;retrieve token
        CP      0BCH            ;TAB( token?
        JP      Z,BUILD         ;yes, don't put blank
        CP      93H             ;REM token?
        JP      Z,BUILD         ;yes, don't put blank
        JR      BLANK           ;no, insert space
;
;>>==>  Conditional blank insert routine
;       Inserts a blank in the output buffer if:
;       1)  we are not processing a REM statement, and
;       2)  we are not processing within quotes,   and
;       3)  there is not a blank there all ready.
;
SPACE   PUSH    AF              ;save (AF)
        LD      A,(QUOTE)       ;get quote value
        OR      A               ;inside quotes?
        JR      NZ,XSPACE       ;yes, no insert
        LD      A,(REMARK)      ;get rem value
        OR      A               ;inside remark?
        JR      NZ,XSPACE       ;yes, no insert
        LD      A,(DE)          ;get last char
        CP      ' '             ;is it a blank?
        JR      Z,XSPACE        ;yes, ok
        INC     DE              ;bump pointer
        LD      A,' '           ;get a blank
        LD      (DE),A          ;and add it
XSPACE  POP     AF              ;retrieve (AF)
        RET
;
;>>==>  At this point, the entire statement is in the
;       primary output buffer.  This section moves the
;       statement to the secondary output buffer and
;       inserts line breaks and indentation as required.
;
PUTLYN  XOR     A               ;zero (A)
        INC     DE              ;mark the end
        LD      (DE),A          ; of the statement
        LD      DE,OUTLYN       ;set output pointer
        LD      HL,LINE         ;set input pointer
        LD      A,(LL)          ;get max line length
        ADD     A,6             ;add length of #
        LD      B,A             ;and put in (B)
PUTNXT  LD      A,(HL)          ;get a byte
        LD      (DE),A          ; and store it
        INC     HL              ;bump in pointer
        INC     DE              ;bump out pointer
        CP      0DH             ;<LF>?
        JR      Z,SPACER        ;yes, insert indent
        OR      A               ;end of input?
        JR      Z,ENDLYN        ;yes, jump out
        DJNZ    PUTNXT          ;no, continue
;
;>>==>  This section will backup and insert a line break
;       in the secondary buffer when the maximum line
;       length has been reached.  If a blank is not found
;       in the last 10 characters stored, the line break
;       is inserted at the maximum line length location.
;
BACKUP  LD      (OLDDE),DE      ;save current (DE)
```

(Continued on next page)

We make C easy . . .

and work!

Eco-C compiler . . . we've got it all.

Whether you're a seasoned professional or just getting started in C, the Ecosoft C compiler has everything you'll ever need.

COMPLETENESS:
Our Eco-C compiler is a complete implementation of C and supports all operators and data types (including long, float and double).

EFFICIENCY:
The compiler generates extremely efficient Z80 code using Zilog's mnemonics. On the benchmarks tested, typically we finished either first or second using substantially less generated code.

PORTABILITY:
The ECO-C library contains over 100 functions that are UNIX V7 compatible, and includes a complete transcendental package. Programs developed with the Eco-C compiler can be moved to virtually any system with little or no change.

EASE OF USE:
The Eco-C compiler includes Microsoft's MACRO 80 assembler, linker, library manager and supporting documentation. The assembler (M80) generates industry-standard REL file output. The linker (L80) is fast and uses only the functions you request in the program. Program development is a snap.
The user's manual is clear, concise and full of useful information. For those of you just getting started with C, we also include a copy of the **C Programming Guide** (Que). This B. Dalton Best Seller has been adopted by a number of leading universities around the country and is included with each compiler. The book is designed to help you learn C from the ground up. We ought to know . . . we wrote the book.
We've made the compiler easy to work with for the professional and beginner alike. Most error messages, for example, tell you in English (not just a number) the line number and character position of the error, what was expected and a page reference to the **Guide** to consult for help if you need it.

PRICE:
We saved the best for last; we've cut the price by $100.00. Now you can buy the Eco-C compiler for only $250.00 (MACRO 80 and the book alone are worth $218.00!). Shop around and we think you agree that the Eco-C compiler is the best value available.
The Eco-C compiler requires a Z80 CPU, CP/M, 54K of free memory and about 240K of disk space (one or two drives). An IBM-PC version will be available in the first quarter of 84. To order your Eco-C compiler, call or write.

ECOSOFT INC.

Ecosoft Inc.
P.O. Box 68602
Indianapolis, IN 46268
(317) 255-6476

VISA MasterCard

TRADEMARKS:
Eco-C (Ecosoft), MACRO 80 (Microsoft), CP/M (Digital Research)

Circle **no. 20** on reader service card.

LISP

FOR THE IBM PERSONAL COMPUTER.

THE PREMIER LANGUAGE OF ARTIFICIAL INTELLIGENCE FOR YOUR IBM PC.

- **DATA TYPES**
  Lists and Symbols
  Unlimited Precision Integers
  Floating Point Numbers
  Character Strings
  Multidimensional Arrays
  Files
  Machine Language Code
- **MEMORY MANAGEMENT**
  Full Memory Space Supported
  Dynamic Allocation
  Compacting Garbage Collector
- **FUNCTION TYPES**
  EXPR/FEXPR/MACRO
  Machine Language Primitives
  Over 190 Primitive Functions
- **IO SUPPORT**
  Multiple Display Windows
  Cursor Control
  All Function Keys Supported
  Read and Splice Macros
  Disk Files
- **POWERFUL ERROR RECOVERY**
- **8087 SUPPORT**
- **COLOR GRAPHICS**
- **LISP LIBRARY**
  Structured Programming Macros
  Editor and Formatter
  Package Support
  Debugging Functions
  .OBJ File Loader
- **RUNS UNDER PC-DOS 1.1 or 2.0**

IQLISP

5¼″ Diskette and Manual	$175.00
Manual Only	$ 30.00

Integral Quality
P.O. Box 31970
Seattle, Washington 98103-0070
(206) 527-2918
Washington State residents add sales tax.
VISA and MASTERCARD accepted.
Shipping included for prepaid orders.

Circle **no. 27** on reader service card.

BASICFMT (Listing Continued, text begins on page 60)

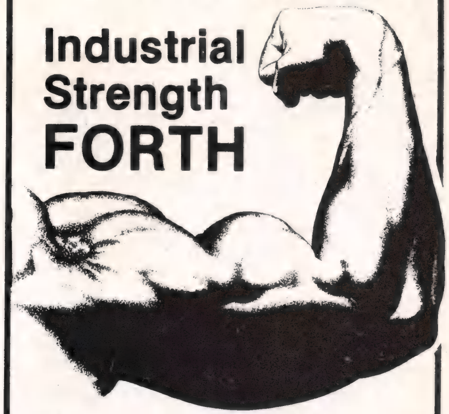
```
02750          LD     (OLDHL),HL      ;and current (HL)
02760          LD     B,0AH           ;limiter value
02770 BACK     DEC    HL              ;unbump pointer
02780          DEC    DE              ;unbump pointer
02790          LD     A,(HL)          ;get character
02800          CP     ' '             ;blank?
02810          JR     Z,BREAK         ;yes, break here
02820          DJNZ   BACK            ;no, try another
02830          LD     HL,(OLDHL)      ;can't break it
02840          LD     DE,(OLDDE)      ; use old breaks
02850          DEC    HL              ;unbump for bump
02860          DEC    DE              ;unbump for bump
02870 BREAK    INC    HL              ;skip blank
02880          INC    DE              ;skip blank
02890          LD     A,0DH           ;insert a <CR>
02900          LD     (DE),A          ; in the line
02910          INC    DE              ;bump pointer
02920 SPACER   LD     A,' '           ;get a blank
02930          LD     B,6             ; and blank count
02940 SPACE2   LD     (DE),A          ;insert a blank
02950          INC    DE              ;bump pointer
02960          DJNZ   SPACE2          ;next?
02970          LD     A,(LL)          ;reset the line
02980          LD     B,A             ; length value
02990          JR     PUTNXT          ;go back for more
03000 OLDHL    DEFW   0               ;storage space
03010 OLDDE    DEFW   0               ;storage space
03020 ;
03030 ;>>==>   The statement has now been stored in the secondary
03040 ;        buffer with line breaks and indentation.  ENDLYN
03050 ;        prints the secondary buffer on the line printer
03060 ;        inserting page eject commands as necessary.
03070 ;
03080 ENDLYN   DEC    DE              ;back up one
03090          LD     A,0DH           ;get a <CR>
03100          LD     (DE),A          ; and insert it
03110          INC    DE              ;bump pointer
03120          XOR    A               ;get a EOL char
03130          LD     (DE),A          ; and mark it
03140          LD     HL,OUTLYN       ;point to output
03150 NEXTLN   CALL   PAGER           ;check page break
03160          CALL   LPRINT          ;print a line
03170          INC    HL              ;bump over <CR>
03180          LD     A,(HL)          ;get next char
03190          OR     A               ;EOL?
03200          JR     NZ,NEXTLN       ;no, get more
03210          RET                    ;all done
03220 ;
03230 ;>>==>   This routine generates carriage returns to
03240 ;        eject a page when the page is full.
03250 ;
03260 PAGER    LD     A,(LINES)       ;get line count
03270          INC    A               ; and add 1
03280          LD     (LINES),A       ; and store back
03290          LD     B,A             ;store in (B)
03300          LD     A,(PL)          ;get page length
03310          CP     B               ;too many lines?
03320          RET    NC              ;no, return
03330          LD     A,1             ;yes, repage
03340          LD     (LINES),A       ;reset counter
03350          LD     A,(PS)          ;get page spacing
```

```
        LD      B,A             ; and store in (B)
        LD      A,0DH           ;<LF> value
LFEED   CALL    LPBYTE          ;feed a line
        DJNZ    LFEED           ;more?
        RET                     ;back for more
;
;>>==>  This routine returns a single byte from the
;       disk buffer to the caller in register A.
;
GETBYT  PUSH    HL              ;save (HL)
        PUSH    DE              ;save (DE)
        LD      HL,(CHRPTR)     ;get byte pointer
        LD      DE,ENDBUF       ;and end of buffer
        INC     HL              ;bump pointer
        CALL    CPHLDE          ;compare them
        JR      Z,GREAD         ;need next sector
GET02   LD      A,(CTLKEY)      ;get control key
        BIT     2,A             ;<BREAK>?
        JP      NZ,EXIT         ;yes, stop
        LD      A,(HL)          ;get the byte
        LD      (CHRPTR),HL     ;store pointer
        POP     DE              ;retrieve (DE)
        POP     HL              ;retrieve (HL)
        RET                     ;send back byte
;       storage for address of last character returned
CHRPTR  DEFW    ENDBUF-1
;       read next sector of the basic program
GREAD   LD      DE,DCB          ;point to FCB
        CALL    READ            ;read a sector
        LD      HL,BUFFER       ;point to buffer
        JR      Z,GET02         ; and return byte
        OR      80H             ;set return code
        CALL    DOSERR          ;display error
        JP      EXIT            ;exit, stage left
;
;================================================
;  BINDEC  -  (HL)=binary  -  (DE)-->output buffer
;  The output buffer must be exactly 5 bytes long.
;================================================
;
BINDEC  LD      (BINA),HL       ;save binary number
        LD      (BIND),DE       ;save beginning of buffer
        INC     DE              ;increment (DE)
        INC     DE              ; to the end
        INC     DE              ;  of the
        INC     DE              ;   buffer
        LD      (BIN4),DE       ;store end of buffer address
        LD      B,5H            ;number of bytes in buffer
        XOR     A               ;initial value for buffer
        LD      HL,(BIND)       ;beginning of buffer
BD1     LD      (HL),A          ;initialize the
        INC     HL              ; buffer to all
        DJNZ    BD1             ;  binary zeros
        LD      HL,(BINA)       ;retrieve the binary number
        LD      A,H             ;get the MSB
        CALL    BD4             ; and convert to decimal
        LD      A,(BINA)        ;get the LSB
        CALL    BD4             ; and convert to decimal
        LD      HL,(BIND)       ;buffer beginning address
        LD      B,5H            ;buffer length
BD2     LD      A,30H           ;convert to ascii value
        ADD     A,(HL)          ;convert digit to ascii
        LD      (HL),A          ; and store back in buffer
        INC     HL              ;increment pointer
        DJNZ    BD2             ;all digits done?
        LD      HL,(BIND)       ;beginning of buffer
```

(Continued on next page)

Multiuser/Multitasking
for 8080, Z80, 8086

Industrial Strength FORTH



TaskFORTH™
The First
Professional Quality
Full Feature FORTH
System at a micro price*

LOADS OF TIME SAVING PROFESSIONAL FEATURES:

- ☆ Unlimited number of tasks
- ☆ Multiple thread dictionary, superfast compilation
- ☆ Novice Programmer Protection Package™
- ☆ Diagnostic tools, quick and simple debugging
- ☆ Starting FORTH, FORTH-79, FORTH-83 compatible
- ☆ Screen and serial editor, easy program generation
- ☆ Hierarchical file system with data base management

* Starter package $250. Full package $395. Single user and commercial licenses available.

If you are an experienced FORTH programmer, this is the one you have been waiting for! If you are a beginning FORTH programmer, this will get you started right, and quickly too!

Available on 8 inch disk under CP/M 2.2 or greater also various 5¼'' formats and other operating systems

FULLY WARRANTIED, DOCUMENTED AND SUPPORTED

master charge DEALER INQUIRES INVITED VISA

Shaw Laboratories, Ltd.
24301 Southland Drive, #216
Hayward, California 94545
(415) 276-5953

Circle no. 53 on reader service card.

```
04020          LD      B,4H            ;buffer length less 1
04030 BD3      LD      A,30H           ;ascii zero
04040          CP      (HL)            ;is buffer char "0"
04050          RET     NZ              ;no, return to caller
04060          LD      A,20H           ;yes, change to a blank
04070          LD      (HL),A          ; and store in buffer
04080          INC     HL              ;increment pointer
04090          DJNZ    BD3             ;loop for all but last char
04100          RET                     ;return to calling routine
04110 BINA     DEFW    0H              ;storage for binary value
04120 BIND     DEFW    0H              ;storage beginning of buffer
04130 BIN4     DEFW    0H              ;storage end of buffer
04140 BD4      LD      B,8H            ;bits per byte
04150 BD5      PUSH    BC              ;save loop counter
04160          PUSH    AF              ;save (AF)
04170          CALL    BD6             ;multiply buffer by 2
04180          POP     AF              ;retrieve the value
04190          RLA                     ;shift bit to carry flag
04200          PUSH    AF              ;save the value again
04210          CALL    C,BD9           ;if bit set, add to buffer
04220          POP     AF              ;retrieve value again
04230          POP     BC              ;restore loop counter
04240          DJNZ    BD5             ;continue for all bits
04250          RET                     ;return to caller
04260 ;
04270 ;>>==>  Multiply the contents of the buffer by 2
04280 ;
04290 BD6      LD      HL,(BIN4)       ;end of buffer address
04300          OR      A               ;reset the carry flag
04310          LD      B,5H            ;length of buffer
04320 BD7      LD      A,(HL)          ;get a digit
04330          ADC     A,A             ;multiply by 2 and add carry
04340          CP      0AH             ;less than 10?
04350          JR      C,BD8           ;yes, skip adjustment
04360          SUB     0AH             ;adjust to less than 10
04370 BD8      LD      (HL),A          ;return to buffer
04380          CCF                     ;switch the carry flag
04390          DEC     HL              ;point to next digit
04400          DJNZ    BD7             ;loop for all digits
04410          RET                     ;return to caller
04420 ;
04430 ;>>==>  Add 1 to the contents of the buffer
04440 ;
04450 BD9      LD      HL,(BIN4)       ;end of buffer address
04460          LD      B,5H            ;length of the buffer
04470 BDA      LD      A,(HL)          ;get a digit
04480          INC     A               ;add 1 to the digit
04490          LD      (HL),A          ; and return it to buffer
04500          CP      0AH             ;digit less than 10?
04510          RET     C               ;yes, return to caller
04520          SUB     0AH             ;no, adjust to decimal
04530          LD      (HL),A          ; and return to buffer
04540          DEC     HL              ;point to next digit
04550          DJNZ    BDA             ;loop to handle carry
04560          RET                     ;return to caller
04570 ;
04580 ;=== DECBIN === Calls: nothing
04590 ;===========================================*
04600 ;======  Convert ascii decimal to binary  ======*
04610 ;===========================================*
04620 ;                                          *
```

```
;  on entry:  (HL) => character string            *
;             terminated by first non-decimal     *
;             character encountered               *
;                                                 *
;  on exit:   (DE) =  binary equivalent           *
;             (HL) => non-decimal character       *
;                                                 *
;=================================================*
;
DECBIN  DEC     HL              ;set initial pointer
        LD      DE,0H           ;zero the contents DE
DB1     INC     HL              ;point to next character
        LD      A,(HL)          ;get next character
        SUB     30H             ;adjust to binary
        RET     C               ;return if character < "0"
        CP      0AH             ;character greater than 9
        RET     NC              ;yes, return to caller
        PUSH    HL              ;save character pointer
        LD      H,D             ;get current value
        LD      L,E             ; from DE register
        ADD     HL,HL           ;old value times 2
        ADD     HL,HL           ;old value times 4
        ADD     HL,DE           ;old value times 5
        ADD     HL,HL           ;old value times 10
        LD      E,A             ;value of new digit
        LD      D,0H            ; to DE register
        ADD     HL,DE           ;add to the old value
        EX      DE,HL           ; and return value to DE
```

(Continued on next page)

BDS C

The *fastest* CP/M-80 C compiler available today

Version 1.5 contains some nifty improvements:

The unscrambled, *comprehensive* new User's Guide comes complete with tutorials, hints, error message explanations and an index.

The *CDB* symbolic debugger is a valuable new tool, written in C and included in *source form*. Debug with it, and *learn* from it.

Hard disk users: You can finally organize your file directories sensibly. During compilation, take advantage of the new path searching ability for all compiler/linker system files. And at run-time, the enhanced file I/O mechanism recognizes user numbers as part of simple filenames, so you can manipulate files located *anywhere* on your system.

BDS C's powerful original features include dynamic overlays, full library and run-time package source code (to allow customized run-time environments, such as for execution in ROM), plenty of both utilitarian and recreational sample programs, *and speed*. BDS C takes less time to compile and link programs than any other C compiler around. And the execution speed of that compiled code is typically lightning fast, as the Sieve of Eratosthenes benchmark illustrates. (See the January 1983 BYTE, pg. 303).

BD Software
P.O. Box 9
Brighton, MA 02135
(617) 782-0836

8" SSSD format, $150
Free shipping on pre-paid orders
Call or write for availability on other disk formats

Circle **no. 4** on reader service card.

FOR $29.95, *MICRO-WYL* WILL WHEN OTHERS WON'T!

You'll know why INFOWORLD rated *Micro-WYL's* performance "Excellent" when you want to insert a file into a file; when you want to print directly from the screen; when you want to see what your text will look like before printing; set tabs where you want them; write programs the easy way; and do many other things that no other word processor can do. *Micro-WYL* is a line editor; use it in conjunction with EMACS derivative word processors and you have the best of all worlds! For $29.95, plus $2.00 postage and handling, have it your way! Sold with the usual Overbeek 30 day money-back guarantee.

LET *Micro-WYL* DO IT ALL TODAY!
Enclosed find check for $31.95 or

charge my Mastercard# ______________ Expires: ____________
Visa# ____________________ Expires: ______________

Check format desired:

__8" SSSD	__Osborne Single Density	__KayPro II
__Superbrain	__Osborne Double Density	__NEC 5"
__Northstar Advantage	__Morrow Micro Decision	__Televideo802
__Northstar Horizon	__Xerox 820 Single Density	__ALTOS 5
__Apple/Softcard	__Xerox 820 Double Density	__Epson

I'm not ready to order now, but send me information about all the affordable programs from Overbeek Enterprises.

Name ______________________________
Address ____________________________
City ____________ State ____________ Zip ____________

OVERBEEK ENTERPRISES, P.O. Box 726D, Elgin, IL 60120
312-697-8420

Micro-WYL—another affordable program from Overbeek Enterprises.

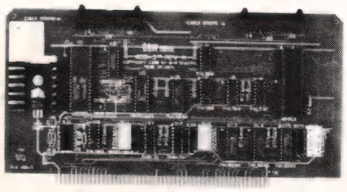
Circle **no. 41** on reader service card.

```
        POP     HL              ;retrieve character pointer
        JR      DB1             ;go to process next char
;
;=== PRINT === Calls: OUTBYT
;===============================*
;  Output a line to the screen  *
;===============================*
;
PRINT   PUSH    AF              ;save (AF) reg
PLOOP   LD      A,(HL)          ;get a byte
        OR      A               ;test or 0H
        JR      Z,LAFIN         ;yes, jump
        CALL    OUTBYT          ;output byte
        INC     HL              ;bump pointer
        JR      PLOOP           ;next byte
LAFIN   POP     AF              ;restore (AF)
        RET                     ;return
;
;=== OUTBYT === Calls: $VDCHAR  (0033H)
;=====================================*
;   Output a character to the screen   *
;=====================================*
;
OUTBYT  PUSH    DE              ;save (DE) reg
        PUSH    IY              ;save (IY) reg
        CALL    33H             ;put the byte
        POP     IY              ;restore (IY)
        POP     DE              ;restore (DE)
        RET                     ;return
;
;=== LPRINT === Calls: LPBYTE
;=====================================*
;  Output a line to the line printer  *
;=====================================*
;
LPRINT  PUSH    AF              ;save (AF) reg
LPNTX   LD      A,(HL)          ;get a byte
        OR      A               ;byte zero?
        JR      Z,LAFINX        ;yes, jump
        CALL    LPBYTE          ;print the byte
        CP      0DH             ;<CR>?
        JR      Z,LAFINX        ;yes, exit
        INC     HL              ;bump pointer
        JR      LPNTX           ;go again
LAFINX  POP     AF              ;restore (AF)
        RET                     ;return
;
;=== LPBYTE === Calls: $PRCHAR (003BH)
;=====================================*
;  Output a byte to the line printer  *
;=====================================*
;
LPBYTE  EXX                     ;save the regs
        LD      HL,37E8H        ;status byte addr
        LD      B,A             ;save character
LP1     LD      A,(HL)          ;load status
        AND     0F0H            ;strip off status
        CP      30H             ;printer on?
        JR      NZ,LP1          ;no, get stuck
        LD      A,B             ;ret character
        CALL    003BH           ;send the byte
        EXX                            ;restore regs
        RET                            ;go back
;
;=== LPSTAT === Calls: nothing
;====================================*
;  Check line printer to see if ready  *
;====================================*
;
LPSTAT  LD      A,(LPPORT)             ;get printer status byte
        AND     0F0H                   ;strip off non-essentials
        CP      30H                    ;is printer ready?
        RET     Z                      ;yes, return to caller
        LD      HL,LPMSG               ;no, get message address
        CALL    PRINT                  ;display message on screen
STUCK   LD      A,(CTLKEY)             ;wait until <ENTER> pressed
        BIT     0,A                    ;<ENTER> pressed?
        JR      Z,STUCK                ;no, wait some more
        JR      LPSTAT                 ;yes, check status again
LPMSG   DEFB    0DH
        DEFM    '>>==> Printer not ready, '
        DEFM    'press <ENTER> to continue.'
        DEFW    00H
;
;=== INPUT (INPUTP) === Calls: INBYTE, OUTBYT, PRINT
;======================================================*
;  Obtain line from the keyboard (with optional prompt)  *
;======================================================*
;                                                        *
;         entry:  (HL) => input buffer                   *
;                 (B)  =  max. number of characters      *
;     (INPUTP):   (DE) => prompt message                 *
;                                                        *
;         exit:   (HL) -  unchanged                      *
;                 (DE) => last character                 *
;                 (C)  -  original contents of (B)       *
;                 (B)  -  actual # of characters         *
;                 (A)  -  termination character          *
;                                                        *
;         note:   The termination character is           *
;                 included in the character string.      *
;                                                        *
;======================================================*
;
QUESTM  DEFM    '? '                   ;question mark prompt
        NOP                            ;prompt terminator
INPUTP  EX      DE,HL                  ;shift prompt addr to HL
        CALL    PRINT                  ;display prompt message
```

(Continued on next page)

THE **488+3** IEEE 488 TO S-100 INTERFACE

IEEE-488



S-100

- Handles all IEEE-488 1975/78 functions
- IEEE 696 (S-100) compatible
- MBASIC subroutines supplied; no BIOS mods required
- 3 parallel ports (8255A-5)
- Industrial quality; burned in and tested
- $375

[Dealer inquiries invited]

D&W DIGITAL
20655 Hathaway Ave.
Hayward, CA 94541
415/ 887-5711

Circle no. 19 on reader service card.

BASICFMT (Listing Continued, text begins on page 60)

```
05990          EX      DE,HL              ;shift buffer address to HL
06000 INPUT    EX      DE,HL              ;shift buffer address to DE
06010          LD      HL,QUESTM          ;point to question mark
06020          CALL    PRINT              ;display question mark
06030          LD      C,B                ;move maximum chars to C
06040          XOR     A                  ;initialize the byte counter
06050          LD      B,A                ; contained in register B
06060          PUSH    DE                 ;shift buffer address
06070          POP     HL                 ; to the HL register
06080          DEC     HL                 ;  and decrement it
06090 ILOOP1   CALL    INBYTE             ;get character from keyboard
06100          LD      (SAVEAZ),A         ; and store for later use
06110          CP      08H                ;is character a backspace?
06120          JR      NZ,STOREZ          ;no, jump to store character
06130          LD      A,B                ;get character count
06140          OR      A                  ;any characters yet?
06150          JR      Z,ILOOP1           ;no, skip backspace
06160          LD      A,' '              ;yes, load a blank
06170          LD      (HL),A             ; and store in buffer
06180          DEC     HL                 ;backup one character
06190          DEC     B                  ; and decrement char counter
06200          LD      A,8H               ;load backspace character
06210          CALL    OUTBYT             ; and display on screen
06220          JR      ILOOP1             ;look for more input
06230 STOREZ   CP      0DH                ;is character <ENTER>?
06240          JR      Z,INRETZ           ;yes, go to return
06250          CP      1H                 ;is character <BREAK>?
06260          JR      Z,INRETZ           ;yes, go to return
06270          LD      A,C                ;get maximum length
06280          CP      B                  ;do we have max. chars?
06290          JR      Z,ILOOP1           ;yes, ignore input
06300          LD      A,(SAVEAZ)         ;no, retrieve character
06310          INC     B                  ;increment byte counter
06320          INC     HL                 ; and address pointer
06330          LD      (HL),A             ;  and store character
06340          CALL    OUTBYT             ;   and display character
06350          JR      ILOOP1             ;go to look for more
06360 SAVEAZ   NOP                        ;character storage area
06370 INRETZ   INC     HL                 ;increment pointer
06380          LD      (HL),A             ; and store last character
06390          EX      DE,HL              ;restore registers
06400          CALL    OUTBYT             ;display last character
06410          RET                        ;return to caller
06420 ;
06430 ;=== INBYTE === Calls: KBSCAN
06440 ;==================================*
06450 ;  Wait for a keyboard character  *
06460 ;==================================*
06470 ;
06480 INBYTE   CALL    KBSCAN             ;look for a byte
06490          OR      A                  ;zero returned?
06500          RET     NZ                 ;no, return
06510          JR      INBYTE             ;yes, look again
06520 ;
06530 ;=== KBSCAN === Calls: $KBCHAR (002BH)
06540 ;==================================*
06550 ;  Scan for a keyboard character  *
06560 ;==================================*
06570 ;
06580 KBSCAN   PUSH    DE                 ;save (DE) reg
06590          PUSH    IY                 ;save (IY) reg
```

```
        CALL    2BH              ;scan keyboard
        POP     IY               ;restore (IY)
        POP     DE               ;restore (DE)
        RET                      ;return
;
;=== CPHLDE === Calls: nothing === equivalent to RST 18H
;==============================================*
;  Compare the contents of registers DE and HL  *
;==============================================*
;
CPHLDE  LD      A,H              ;get MSB number 1
        SUB     D                ;compare MSB's
        RET     NZ               ;return if not equal
        LD      A,L              ;get LSB number 1
        SUB     E                ;compare LSB's
        RET                      ;return to caller
;
;    ***************************************************
;    *  Data storage areas, buffers, and control blocks  *
;    ***************************************************
;
LINES   NOP             ;number of lines printed value
QUOTE   NOP             ;quote processing logical variable
REMARK  NOP             ;remark processing logical variable
LL      DEFB    79      ;line length (less 6)
PL      DEFB    60      ;page length (number of lines)
PS      DEFB    6       ;page spacing (lines between pages)
DCB     DEFS    32      ;File Control Block (FCB)
BUFFER  DEFS    256     ;Input file buffer
ENDBUF  EQU     $       ;end of input file buffer label
OUTLYN  DEFS    512     ;secondary output buffer
LINE    EQU     $       ;primary output buffer
ENDPGM  EQU     $       ;end of program label
CORE    EQU     ENDPGM-BASFMT   ;core used calculation
        END     BASFMT
```

End Listing

NOW FOR THE IBM PC + COMPATIBLES

WINDOWS FOR C™

SCREEN MANAGEMENT TOOL FOR C PROGRAMMERS

A COMPLETE WINDOW DISPLAY SYSTEM

- Unlimited windows and text files
- Instant screen changes
- Overlay & restore windows
- Change attributes
- Horizontal and vertical scrolling
- Variable text margins
- Word wrap option
- ASCII file handling

SIMPLIFY • IMPROVE

- Menus
- Help files
- Data screens
- Text handling

ALL DISPLAYS

C SOURCE MODULES FOR
menus, cursor control, ASCII file display, text-mode bar graphs;
+ complete building block subroutines

VERSATILE • FULLY DOCUMENTED

Available For: Lattice C, C86, Microsoft C, DeSmet C (MS or PC DOS) (please specify)

Introductory Special $119.95
($150 after May 31)
Demo disk & manual $30
(Applies toward purchase)

A PROFESSIONAL SOFTWARE TOOL FROM
CREATIVE SOLUTIONS
21 Elm Ave., Box D4, Richford, VT 05476

For Order or Information: 802-848-7738
Master Card & Visa Accepted
Shipping $2.50
VT residents add 4% tax

Circle no. 12 on reader service card.

SOFTWARE REVIEWS

microSUB:MATH

Company: foehn consulting, P. O. Box 5123, Klamath Falls, OR 97601

Computer: Microsoft Fortran-80 and Cromemco Fortran IV (CP/M-80 only), Supersoft SSS Fortran (CP/M-80 and MSDOS), and IBM Fortran 77

Price: $250.00 ($25.00, manual only)

Circle Reader Service No. 133

Reviewed by Elbert F. Hinson

microSUB:MATH is a rich collection of Fortran numerical-methods subroutines for use in the scientific and engineering fields. These routines are meant for serious Fortran programmers. The subroutines in this package provide many of the mainframe computer Fortran library routines for use on a microcomputer.

The package I received was for Microsoft Fortran-80 on a SSSD, 8-inch CP/M 2.2 disk. The routines are also available in almost any popular 5¼-inch disk format and are available for SuperSoft Fortran and Cromemco Fortran. By the time you read this article, a version should be available for the IBM PC.

The software is organized into six general areas. Two libraries, one for single precision and another for double precision, are provided for each area. The six areas are functions, interpolation, integration, matrix/linear systems, polynomials/nonlinear equations, and differential equations. Table 1 (at right) contains a complete description of the contents of each single-precision library. The double-precision libraries contain the same routines but provide the accuracy of double precision. The double-precision library names and subroutine names are prefixed with a D. For example, the library DFUNC is for double-precision functions and function DSINH is the double-precision hyperbolic sine.

Documentation

The manual supplied has a table of contents, an index, and tabs for each chapter. The introductory section gives an overall view of the available routines and how to use them. The method of linking the library routines with your loader is covered in sufficient detail for anyone familiar with Fortran. This is not a tutorial, so some knowledge of Fortran is expected if you wish to use these library routines.

Part of the introductory section is devoted to building custom libraries. This details which of the microSUB:MATH routines call other routines in the library and in what order the routines must be loaded in a custom library. Each of the six library sections has an individual write-up of each routine in the library.

The subroutine documentation is clear and easy to use. The manual provides the Fortran calling sequence, an explanation of the input and output parameters, an explanation of the method used and a sample program for each subroutine in the library. There are appendices for quick reference and test program results.

A set of test programs is provided on the distribution disk that exercises every subroutine in every library. Results of the test programs are provided in the manual for single precision but not double precision.

Performance

I tested a significant number of the subroutines but did not have time to test

Library	Subroutine	Purpose
FUNC (Functions)	SINH, COSH, TANH	Hyperbolic trigonometry
	ASINH, ACOSH, ATANH	
	CM, CD, CEXP, CLOG, CPWR	Complex math – multiply, divide, exponential, natural log, and raise to power
	CSIN, CCOS, CTAN	Complex trigonometry
	PR, RP	Polar to rectangular and rectangular to polar transform
	RJN, RJV	Bessel functions, integer and real orders
	ERF	Error function
	GAMMA	Gamma function
	SFRES, CFRES	Fresnel integrals, S(x) and C(x)
	SI, CI	Sine and cosine integrals
	CNST	Constants – PI, Euler's, and computer accuracy
INTP (Interpolation)	LIN	Linear interpolation
	LAGR	Lagrange interpolation
	UDVTB, UDVIN	Table for divided difference and solution for unequally spaced points
	EDVTB, EDVIN	Table for divided difference and solution for equally spaced points
	SPLTB, SPLIN	Second derivative table for cubic spline and cubic spline interpolation
	LAGDB	Double Lagrange interpolation
INTG (Integration)	SIMP	Simpson's
	ASMP	Adaptive Simpson's
	ROMB	Romberg
	GAUS1	Low order Gauss-Legendre
	GAUS2	High order Gauss-Legendre
MATX (Matrix/ Linear Systems)	RLU, SLU	LU decomposition and solution by Gauss elimination
	SOER, SOES	Gauss elimination with scaling and partial pivoting and solution from elimination
	GJOR	Gauss Jordan elimination
	MTM	Matrix multiplication

Table 1.

every one of them. I compared the results of most of the functions for random values against the CRC math tables. Function results appeared good for six digits of precision. Double precision should provide more accuracy if it is needed.

I ran parallel tests of many of the Matrix, Integration and Polynomial subroutines against the programs in the book *Fortran Programs for Scientists and Engineers* by Alan R. Miller. The results were the same for at least five digits in the single-precision versions. Although I did not specifically benchmark the results, the routines appeared to execute at about the same speed. I found it very easy to substitute the subroutines from the libraries for the subroutines in Mr. Miller's book. It took an awful lot of my time and typing to build the same thing that is contained in the library.

Support

I talked to Mr. Hugh Currin of foehn consulting about the package and some of my wish list. The results of the double-precision test programs should be included in later releases, but this present omission should provide no real problems to the user. I asked Mr. Currin if he had considered including the transcendental functions tangent, arcsin, and arccos that Microsoft left out of their library. He indicated that they could be included in a later release.

Conclusions

I highly recommend this package for any scientist or engineer who would like the power of mainframe subroutine packages on a microcomputer. There is a significant amount of scientific work that can be done on a dedicated microcomputer instead of fighting a busy mainframe. This package is very easy to use and contains a very powerful set of numerical subroutines for microcomputers.

(Continued from previous page)

Library	Subroutine	Purpose
	MINV	Matrix inversion
	JCBI	Jacobi, eigenvalues and -vectors of real symmetric matrix
	HQR1	HQRI eigenvalues of a real symmetric matrix
	INVIT	Eigenvector of a real symmetric matrix
	HQR2	HQRI eigenvalues of a general real matrix
	HQR3	HGRI eigenvalues and -vectors of general real matrix
POLY (Polynomials/ Nonlinear Equations)		Real root of an equation:
	HINT	Half interval method
	BRENT	Brent's method
	NEWT	Newton's method
	NEWTS	Nonlinear system of equations (Newton's method)
	JACM	Jacobian matrix (used in NEWTS)
		Orthogonal Polynomials:
	CHB1, CHB2	Chebyshev, first and second kind
	HERM	Hermite
	RJACB	Jacobi
	RLAGR	Generalized Laguerre
	RLEGN	Legendre
		Roots of Polynomials:
	PLYR1	Second, third, and fourth order
	PLYR2	One polynomial root (Muller's method)
	POLY	Polynomial evaluation
	PLYR3	Roots of polynomial (Muller's method)
DIFEQ (Differential Equations)	RGK	Runge-Kutta, fourth order
	ADMLT	Adams-Moulton, fourth order
	RKF	Runge-Kutta-Fehlberg variable step size

Table 1.

METAFILE Application Development System

Version 7.0 (Major_Updates. Minor_Updates)
Company: Sensor-Based Systems, 15 East Second Street, Chatfield, MN 55923-1297, (507) 867-4440
Computer: IBM PC or XT. IBM PCDOS version 1.1 or 2.0. Monochrome or color monitor. Minimum 128K RAM; 640K supported. One 2s/2d floppy disk drive plus one other storage unit (floppy or hard disk). Asynchronous communication adapter - RS232C. Printers supported: IBM Epson, NEC 3550, Dataproducts M200, Okidata 84
Price:$1050 (minor updates $25; major updates $350)
Circle Reader Service No. 135
Reviewed by Morton F. Kaplon

The vendor's release describes METAFILE as follows:

"The METAFILE Software System creates a development and operational environment that includes integrated facilities for high-level programming, word processing, data base management, modeling, report generation, and communications."

This does indeed reflect the capabilities contained within METAFILE. The dictionary lists a variety of meanings for the prefix "meta," but clearly the one intended here is that of "beyond" or "transcending." And METAFILE certainly does extend the usual concepts associated with filing in its most general sense.

The potential users of this package could easily include a commercial enterprise requiring sophisticated management support, an applications programmer writing in a high-level language (METAFILE, in this case), and a stand-alone individual who likes to have a lot of capability at his or her fingertips. An example, which demonstrates both the program's integration and its power, is the ability to incorporate extensive numerical or logical calculations within a word processing environment and to store those calculations in either an evaluated or an unevaluated form so that the input can be changed, rather like a spreadsheet (though this is certainly not a spreadsheet), to obtain an output reflecting the change.

In order best to understand METAFILE's structure, we have characterized it as an operating system, called METAFILE ASSISTANT, that controls access to a series of integrated and interacting programs. These programs include: Record Management; Text Management; Text Print Control; and METAFILE Command Language. These terms are from the manual. In more familar language the programs are, respectively: Relational Data Base; Full Screen Editor; Text Formatter; and METAFILE Interpreter.

METAFILE employs two distinct operational screens. One, which we will refer to as the COMMAND mode screen, can be considered as METAFILE's operating system screen, and the other, which we will call the EDIT mode screen, is that of the text editor.

COMMAND Mode

When you call METAFILE from DOS, the first screen presented prompts for a user ID (optional) and the date/time if not entered via a clock/calendar on boot to DOS; it then asks you to press Enter. The user ID is a METAFILE built-in function and could be easily adapted, using their programming language, to control operator file access. The next screen displays the COMMAND mode of METAFILE (Figure 1, below).

The screen is 80 columns wide by 24 rows deep. The failure to use the 25th row, available on the IBM PC, is probably related to the program's origins from the "Vector 3." These origins are reflected also in the way the function keys on the keyboard are ignored, using instead various Ctrl-letter combinations. These aspects show a weakness in design that Sensor-Based Systems states will be remedied in their next release.

The 23rd row is used for status information, and the 24th row is used for command area entry and for messages. The 22nd row is held blank to allow separation between the display area and the status lines, thus leaving a usable display area of 21 lines. Both the status area and that containing the METAFILE ASSISTANT are in reverse video.

The METAFILE ASSISTANT controls access to a set of integrated programs. METAFILE is essentially a menu-driven program, choices being made by positioning the cursor on an option and pressing the Return key; depending on the option selected, further menus may be presented for additional selection. Some commands can be entered directly by typing the appropriate command, thus bypassing some intermediate menu structure.

Perhaps the two most-used keys with METAFILE are Esc and ~, the latter not requiring the Shift key. ~ is referred to as the ATTN key, and when pressed it presents in the display area the relevant menu for the program in operation *or* the METAFILE ASSISTANT menu. The Esc key is implemented in the same way as in VisiCalc, Lotus 1-2-3, and many other programs: it takes you back one level in your calling sequence.

CALCULATOR

Of the five principal options, CALCULATOR offers the most immediate access. This option positions you directly in a screen insert, in reverse video and of the same contour as METAFILE ASSISTANT, from which you may perform a variety of numerical and logical calculations using the METAFILE Interpreter.

In the CALCULATOR mode, you can carry out any of the allowed METAFILE operations using numbers, strings, defined variables, and built-in functions for its operations. Calculations are carried out in floating decimal to 20 places. Included in the operations are add, subtract, multiply, divide, raise to power (**), operations on "dates," and the standard set of relational operations, for both numbers and strings, yielding a 1 if true or a 0 if false. The logical AND, OR conjunctions are included.

Built-in functions include natural logarithms, absolute value, random numbers, a variety of "date/time" functions, a function returning a one-byte binary equivalent of the value of its argument (this is used for printer control addressing), and a host of others related principally to data base structures. Arithmetic is performed strictly from left to right, and a generous use of ( ) is suggested to ensure that CALCULATOR does what is intended.

ACTIVE CONTEXT

The ACTIVE CONTEXT option offers a menu enabling selection from a listing of submenus. These summarize the active resources in METAFILE. They include: inter alia; the status of parameters established in the text editor, such as margins, search and translate values, page formats, printer status, etc.; structure of record files; listing of variables established with memory used; name of current drive; active files; and a host of other parameters defining the current status of the system. ACTIVE CONTEXT is presented in the same format as METAFILE ASSISTANT.

LOOKASIDE

The LOOKASIDE option is a particularly useful utility. Essentially, LOOKASIDE is a split-screen facility, but in METAFILE's realization, the alternate view is presented in the same reverse video setaside contour as the METAFILE ASSISTANT. Selecting LOOKASIDE presents you with a submenu of three choices: EDIT, UPDATE, and VIEW, as well as the option of typing in a command, as is offered in METAFILE ASSISTANT.

It is useful to think of LOOKASIDE as a second (albeit somewhat less enriched) command screen from which to exercise METAFILE's capabilities. Thus, selecting one of the options can lead to further menu choices, as from METAFILE ASSISTANT, or entering a command directly on LOOKASIDE's com-

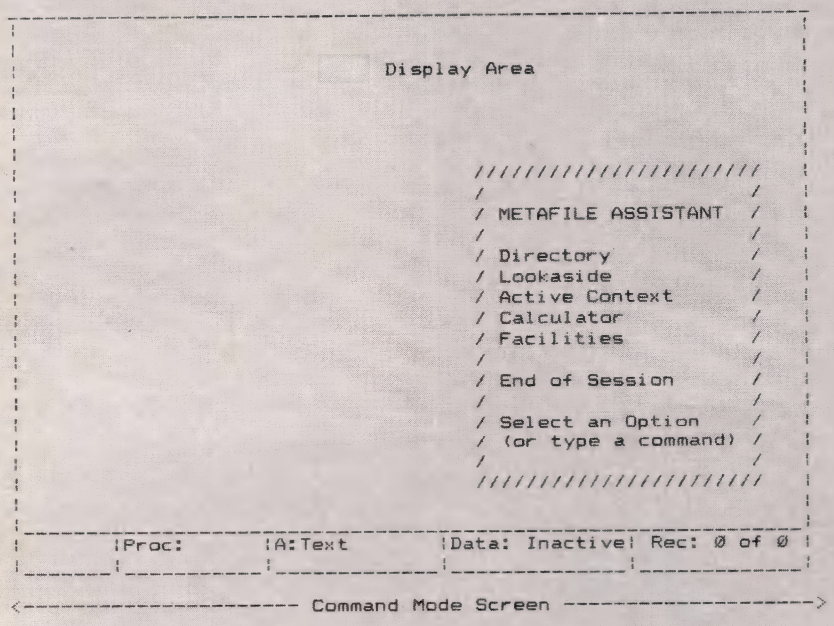


Figure 1.

mand line can enable you to move more directly to the task.

Selecting the EDIT option allows you to select a text for editing. Choosing the VIEW option allows a passive interaction with a file or record (like DOS's "type"). Choosing the UPDATE option allows operations to be performed with the data base. Other options may be entered directly. In LOOKASIDE, you have the same full power of the utilities available as if they had been called from the COMMAND screen.

Used in conjunction with the regular EDIT mode, this option allows the equivalent of "copying" from the LOOKASIDE file into the current text buffer by utilizing the include facility of METAFILE's editor. This allows you to look at what you may want to include in LOOKASIDE, edit it, write it out to file, and then read it back in, into the active text buffer.

DIRECTORY

The DIRECTORY option is concerned both with the management of the storage devices and with the active text buffer. Since METAFILE considers certain file types as reserved (e.g., those dealing with procedures, descriptions, and forms) and distinct from text files, this option offers the viewer the choice of a file listing by drive and type (and text is considered a type) or, under General Directory, by drive without restriction. Optionally, the current text buffer can be chosen. Once a file is selected, the user is presented the choices of EDIT, VIEW, or PRINT. Finally, data is presented on storage space availability and entry utilization. Some of the submenus of DIRECTORY are also available from other menus.

FACILITIES

The FACILITIES menu offers the largest number of submenus: nine, plus the usual choice of typing in a command. Each of these options in turn references further submenus. If METAFILE has an operational core, it is contained here. Included is access to basic operating system commands dealing with file management, such as storing, activating, erasing, renaming, transferring from one storage device to another, and establishing communications with other METAFILE systems via an "on line" (local network). Access to the editor is also available here.

Of particular importance is entry to the large resource of commands dealing with the data base; these include commands relating to its definition, maintenance, and interrogation, as well as the structures relating to form preparation and report generation. An additional option available relates to the procedures to "import" data base files from "foreign" formats into METAFILE.

EDIT Mode

The EDIT mode contains the (full) screen editor. The display area is 80 columns wide and 21 rows deep. Row 22 is blank, and rows 23 and 24 are used for status and prompts, respectively (row 25 is not utilized). The status line displays (in reverse video) the filename; line and column number of the cursor with respect to the upper left corner of the complete text; line and column number of the cursor with respect to the current screen (frame); margin status; mode status (wordwrap, insert or overtype, etc.); and percent of text space filled. Horizontal scrolling up to 250 columns occurs if wordwrap is not enabled at a column less than 80.

METAFILE's EDIT buffer is maintained entirely in memory and has a maximum capacity of about 63K. Due caution is required in movement between the EDIT and COMMAND modes so that the active EDIT file is not lost; METAFILE has inserted prompts to ensure that the buffer is not inadvertently lost.

The EDIT mode is used for a variety of purposes in addition to straight text editing. On one level it can be viewed as the editor for the METAFILE command language. In EDIT you can write a METAFILE procedure (program) and run it (note the similarity to BASIC). The editor is also used to create forms for data entry and to format reports.

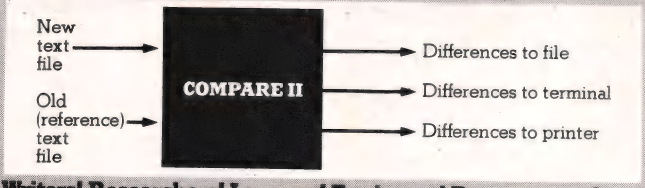
Access to the various editing commands is via the ATTN (~) key where a menu is presented. Selection can be made by positioning the cursor on the command and pressing Enter or by typing the first letter of the command. A help facility is available. The editor is crisp in its response to commands. In fact, the only slow response was that of its type-ahead buffer, principally when scrolling beyond column 80. In that case, it was easily fillable, to which it responds with the standard IBM "beep."

As a word processor, EDIT is quite good, comparing quite favorably with many stand-alone programs. It has some very good features that many do not have and lacks some that would make it even better. One of its nicer features is the automatic reformatting (if wordwrap is enabled) of text in the insertion mode. A negative feature is the lack of any overall reformatting capability. Once the margins are set, you cannot return, change them, and reformat. It has a reasonable set of cursor and text controls: move, copy,

No more lost edit changes!

COMPARE II

High performance differential text analyzer!



Writers! Researchers! Lawyers! Engineers! Programmers! Use highly rated COMPARE II. Cut text analysis from hours to minutes!

- ▶ PC-DOS, CP/M-86 or CP/M 2.2
- ▶ Scans by word or by line
- ▶ Fast New Algorithm, No file restrictions
- ▶ You can customize for word processor, printer width, file defaults, specific work flow, computer languages, different highlighting techniques
- ▶ Clear commands, Numerous formatting options
- ▶ Can generate new document with change bars

Specify When Ordering; Operating System, Computer Type and Disk Format. Free brochure and *nearly* free demo disk available.

COMPARE II **$145.00**
Demo Disk (credits to purchase) $12.95

SOLUTION TECHNOLOGY, INC.

"We **Deliver** Productivity"

1499 Palmetto Park Road
Suite 218
Boca Raton, FL 33432
305/368-6228

PC-DOS and CP/M are trademarks of IBM and Digital Research respectively.

Check or COD. Florida residents add 5% sales tax. Dealer and Distributor inquiries welcome.

Circle **no. 60** on reader service card.

delete line and text block, search and replace, and the ability to insert files into text, to print blocks of text, or to save them to disk.

Text Formatting

Extensive formatting commands allow headers, footers, centering, protection of text block on printing, multiple print copies, printing of selected pages, automatic page numbering and date inclusion, right justifying, and others. Practically any METAFILE command can be included in the print formatting, and printing can be done very easily from within METAFILE. The print control commands are included in the text (on separate lines) using *%command* at column 1, where *command* is any valid print command or any METAFILE command. Comments may be included by following the % symbol with a semicolon, followed by the comment.

Special Features

It is worth considering some of the unusual facilities of METAFILE. One of these is the ability to include operations within a text file. Any allowed METAFILE command or active variable can be accessed in the EDIT mode by placing the variable name or the METAFILE command within { }. The editor has a command named EVALUATE that displays the value of the variable or of the operation contained within { }. EVALUATE toggles between evaluated and unevaluated displays.

For printing or saving text to file, one can choose either the evaluated or the unevaluated mode. Since the variables defining record parameters are included here, it is clear that this offers an exceptionally rich tool in the generation of forms, reports, and letters, utilizing elements from the data base. It is this feature that makes mail merge and the printing of personalized letters extremely easy from METAFILE.

Consider a simple example: an array variable X(i), where each element is a name. A given name is referenced as {X(i)}. On calling EVALUATE, a name is displayed. For the case where the X(i) are numbers, if {X(2) * X(3) - X(4)} is entered in text, on calling the EVALUATE command, the substitutions are made and the formula evaluated.

As with many things, there is a price to pay. The METAFILE editor counts characters in the unevaluated mode. Thus, if {A} (containing on evaluation "Mr. Jones") is within the wordwrap margin in the unevaluated mode but extends beyond it on evaluation, wordwrap does not occur; on printing the string, "Mr. Jones" will extend beyond the wordwrap margin. However, the failure to correctly determine the character count on the unevaluated { } has more serious consequences for embedding printer controls in text, which may be required since METAFILE's current printer support is rather minimal.

The command to embed a control is {Code (#)}, where # is the decimal value of a number in the print control sequence. Here the embedded format dominates in the determination of the column number for wordwrap, and it will always occur before the evaluated location. You are also offered the option of implementing printer control codes by text formatting; the format is: "%CONTROL *formula*" where *formula* is as above without the { }. This will not affect wordwrap, but it does not allow in text embedded controls.

It does not require too much imagination to see that it is possible to produce formatted spreadsheets using the { } utility. It is not a substitute for a true spreadsheet, since to change the value of a variable requires that the METAFILE "SET" command be used (equivalent to BASIC's LET VARX = xxx), but it does offer a very useful tool.

METAFILE's procedures for creating menus are particularly easy to implement. This is done by using [ ]'s to contain METAFILE commands or the names of executable METAFILE procedures as the options in the menu. An active text, with these choices enclosed within [ ]'s, is the basis for user-defined menus.

The menu and its options are formatted within the EDIT mode under a user-selectable filename with type MEN. A METAFILE procedure of the same filename but of type PRO is written (the manual lists the procedure; it is a mere seven short lines). The menu is called from a command line by typing in its filename. Under control of this program, the cursor keys control movement between options. Return selects an option, and when it is concluded the menu is presented again for further selection; pressing the Esc key concludes the menu session.

The options, as noted, can include METAFILE procedures so that the menu allows selection of user-written procedures to accomplish specified tasks. Since METAFILE allows a procedure to call others as subprocedures, it is clear that the language offers powerful capabilities for user definition and employment.

System Parameters

METAFILE has the implementation limitations listed in Table I (below) METAFILE admits three types of data elements: character (C), numeric (N), and data (D), with implementation restrictions noted in the table.

In addition to restrictions on implementation, another very important aspect of any data base is its operational performance in interrogating files. How long does it take to sort a file of 10,000 records on a single key? With qualifications? And so forth. Unfortunately, these questions are not addressed in the manual, and this reviewer has certainly not had the time to create such a base and test its performance. METAFILE, however, is not unique in failing to address these problems in its documentation.

It is not really meaningful to carry out a test on a small data base since it does not yield meaningful results for the potential user with a data base of significant proportions. This aspect poses a serious problem for any prospective user of any data base program, and it is not one that has been appropriately addressed. This is an area that calls for clarification for all such programs.

1.	Maximum number of characters in a filename	8
2.	Maximum number of characters in a file type	3
3.	Maximum length of a record (characters)	1000
4.	Maximum number of records in a file	32000
5.	Maximum size of a text (approx. pages)	15
6.	Maximum length of a text line (characters)	250
7.	Maximum length of an item (characters)	235
8.	Maximum precision of a number (digits)	20
9.	Double quote (") not allowed in record item.	
10.	Backslash (\) not allowed in text line.	
11.	File and item names may contain only alpha numeric characters and underscores (_).	
12.	Date must lie between 01/01/1776 and 12/31/9999	

Table 1.
METAFILE Restrictions

Installation

The package contains three Reference Guides, an Installation Guide, a "Configuration Device," and two double-sided, double-density disks. The program disk has 317,440 bytes of operational files, and the demo disk has 284,672 bytes of files. The program disk is so packed that if you are operating under DOS 1.1 you cannot get any DOS system files on it. Under DOS 2.0 you can just barely make a bootable disk with the system files, but when you go to enter your printer configuration (even as small a file as that), you obtain a "disk full" message.

Thus, for practical purposes a separate disk to boot the system is required, and you can then include such useful utilities as a clock calendar (whose input METAFILE automatically accepts); ProKey, which works quite well; and a RAM disk, which is quite useful. In exploring METAFILE, I used a boot disk formatted as indicated above, with an autoexec.bat file that, among other things, copied the program files in drive b: to the RAM disk (after a prompt to install in drive b: a disk with program files). I then put a data disk in drive b: (again on prompt) and was set to go.

METAFILE is copy protected via hardware. The disks per se are not protected, but to utilize them effectively you must connect the METAFILE "Configuration Device" to the COM 1 serial port. This device is a short cable having female and male connectors with a "black box" in between. The connector choices allow you to reconnect your modem (or any other device that may have been using the port) onto the cable and operate normally.

I tried my modem in that configuration and it worked without problems. Without the special cable connection, METAFILE can function but with only minimal memory (about 12 percent of system capability) and as such is useful only for demonstration. The device operates to restrict the system disk or its copies to a single machine for full utilization. There are the usual warranties to replace either the system disk or the "Configuration Device" within 90 days if either malfunctions.

Since METAFILE will operate with 128K RAM and the program files take up almost 320K, it is clear that many calls are going to be made to the program disk. If you do not have a hard disk, operating from a floppy can be rather slow; you could, however, use a RAM disk, which will afford some increase of speed even with a hard disk. METAFILE's instructions must be followed very closely for operation with a RAM disk. To operate with RAM, METAFILE is called from DOS with the command: metafile , memory:*xxx*K, where *xxx* is the amount of memory being reserved for METAFILE in units of 1000.

Since METAFILE's capabilities and capacities are affected by the amount of memory available (delineated for the user in a table in the Installation Guide), reasonable care should be exercised in utilizing a RAM disk. Since at least 320K is needed for the RAM disk, a user contemplating that approach should, by and large, have a totally configured system with respect to memory. Of course if you operate from a hard disk, you have more options available.

In any event, METAFILE demonstrates some peculiarities with respect to RAM disks. I use two different programs. One utilizes the RAM disk driver contained in DOS 2.0, modified for higher capacity, and the other is Tall Trees' JFORMAT. METAFILE seems to accept the former with great tolerance, behaving well even when I forget to call it as indicated above. On the other hand, with JFORMAT, unless it is called in precisely the specified format, the system hangs up; if called correctly, it performs flawlessly.

There is one operational oddity, at least in the system I have, that is independent of the use of RAM disks or of any other peripheral hardware or software. After once calling METAFILE and closing a session to return to DOS, the DOS underlined blinking cursor is lost and does not return until the system is rebooted (warm does it OK); otherwise, DOS performs normally. I must admit that loss is a little disconcerting, but it is not disfunctional.

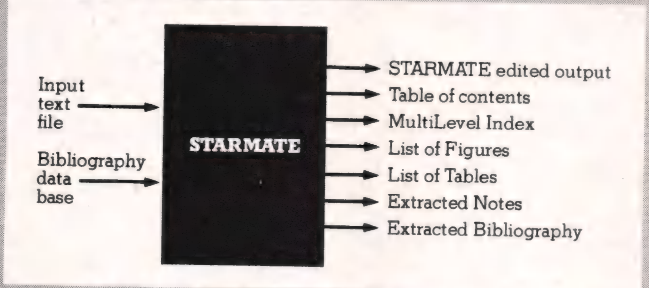
Documentation

The Installation Guide tells you how to use the demonstration facility (the demo disk). I find demos hard to evalu-

Get more than WordIndex for less $!

STARMATE

Users get high speed document finishing for WordStar with STARMATE under PC-DOS, CP/M-86, or CP/M 2.2.



Writers! Researchers! Engineers! Cut document makeup time from hours to minutes!

- ▶ Multi-Level Table of Contents
- ▶ Mulit-Level Index
- ▶ Lists of Tables and Figures
- ▶ Numbers Paragraphs (1.2, 1.3, etc)
- ▶ Prepares Footnotes
- ▶ Prepares Bibliography
- ▶ Clear commands, Numerous formatting options
- ▶ Reads documents with nested files

Specify When Ordering; Operating System, Computer Type and Disk Format. Free brochure and *nearly* free demo disk available.

STARMATE
(Special Introductory Price) **$145.00**
Demo All Disk (credits to purchase)..................... $12.95

SOLUTION
TECHNOLOGY, INC.

"We **Deliver** Productivity"

1499 Palmetto Park Road
Suite 218
Boca Raton, FL 33432
305/368-6228

WordStar, PC-DOS and CP/M are trademarks of MicroPro, IBM and Digital Research respectively.

Check or COD, Florida residents add 5% sales tax. Dealer and Distributor inquiries welcome.

Circle **no. 59** on reader service card.

ate. It is my belief the reasonably educated user is probably going to have more difficulty than a novice, principally because he or she just does not follow them with sufficient attention. On occasion I have asked my children or my wife to react to a demo disk of some program, and they have invariably been more satisfied with it than I was.

With that caveat, I was not overly impressed with the demo disk. It did introduce you to the various aspects of what the system can do, asked you to create some records, to enter some text, to correct it, to print it, to do some finds and replaces, to sort a file, etc. But I do not really believe that someone who does not have some understanding of what files, records, and fields mean can really use the demo and get very much out of it.

METAFILE is supported by three separate reference documents: spiral bound, 8.5 x 11 inches in size, with heavy paper covers.

Volumes 1 and 2 are labelled as User's Guides. Volume 1 is dedicated to A. Word/Text Processing and B. Record Management; Volume 2 to C. Introduction to Programming, D. Applications Design, and E. Glossary/Index. Volume 3, about twice the thickness of either 1 or 2, is labelled Reference Manual. These manuals have many faults. One of the most nettlesome is that they are "dirty." I found after using them even briefly that my work area was full of lint generated by the manuals and I was constantly vacuuming it.

The manuals are by far the weakest part of the package and do not, in fact, do it justice. Operationally, they are not organized for ready reference: there are no index tabs to allow quick selection of text sections, and while there are tables of contents for each of the major subject areas, the indexes are associated only with the glossary and with the Reference Manual. The latter two indexes are reasonably well presented, each organized alphabetically with a description and a page reference for each item, but there is a fair degree of redundancy as well as missing items. Included with the reference documents are a METAFILE User Reference Card (8.5 x 3.5-inch folder) and a Function Key Specification card of the same dimension.

It is very cumbersome, particularly when finding your way through a new and rather complicated package, to have to shuffle three separate documents, none particularly well designed for reference. The vendors do indicate that new documentation and a new format (the IBM notebook and slipcase style) is in preparation for future release. It is sorely needed.

The first two User's Guides are hands-on tutorials. Volume 1 takes you through a subset of the capabilities of the word processor and includes a tutorial on record management using a hypothetical personnel office as a case study. Volume 2 includes a tutorial on the programming facilities of METAFILE, again using a case study based on a set of employee records, and a final set of case studies dealing with the use of advanced commands and facilities, application guidelines, prototyping techniques, testing, debugging, and documentation.

The last three tutorials contain many useful examples and METAFILE programs that can be employed by the user, demonstrating such diverse aspects as mail merge, label addressing, form letter preparation, report documentation, and "converting" documents to METAFILE from foreign formats. The conversion presented is for record formats only, and I was unable to find any discussion dealing with the "importation" of text files. Some experimentation indicated that the METAFILE editor will accept only the purest of ASCII text files. Since a reformat capability is not included, the ability to import text files is relatively moot.

Displaying the hierarchy of a command structure, particularly with a program so richly structured as METAFILE, in a readily comprehensible fashion is important. This is an area where the documentation is also quite weak. Although the structure is presented, it is not done systematically nor in a way that gives an overall perspective; as a result, it is not useful for orientation or reference. The user is unable to discern the relationships between various units without an undue and uncalled for effort on his or her part. A tree diagram would have been most useful.

Printer Support

A few comments on printer support are in order. The documentation states that the IBM Epson is supported and it means just that: the bare-bones Epson without Graftrax. This is not made very clear, and certain printer commands indicated in the manuals are not realizable with any Epson other than the bare-bones IBM. It turns out that it is possible to embed printer controls in text, as well as in other formats, due to the enormous capabilities of the program, but it is almost impossible to discern from the manuals how to do this. I had to resort to the vendor for information.

Sensor-Based Systems is very responsive to requests for assistance and was able to head me in the right direction. As a result, I found it possible to utilize the full text print capabilities of the Epson MX 100 (and, I would judge, any other printer) since decimal sequences can be sent to the printer from within text files (with care, as noted above) or from formatting commands. Printer support is certainly one of the weakest operational aspects of the program.

Any serious user of METAFILE in substantive proportions will almost certainly need print spool capabilities. A print spooler is not included, and given the memory requirements of METAFILE, the most efficient way to obtain that would be with a stand-alone spooler.

Use of the Keyboard

Although a card indicates the METAFILE function key specifications, the word "function" is used operationally. The IBM function keypad is not used at all. In addition to a reasonably standard use of the cursor pad, 16 METAFILE-dedicated commands are effected by Ctrl-letter combinations. Included in the 16 are two cursor-control commands.

Some of the letters in the Ctrl-letter combinations have a relation to the command but many do not; it seems to me a significant improvement would be to assign to the function keys their related combinations. In fact, I reassigned them with ProKey so that they coincided, insofar as possible, with similar assignments I use on other programs (one of the great advantages of products like ProKey is cross-program standardization). When I commented to the vendors on their failure to use the function keys, their response indicated that future releases would include a built-in macro facility that would allow the user that option, among others. This would be an enormous enhancement indeed.

METAFILE Programming Language

METAFILE's programming language encompasses a wide range of capabilities. It is primarily a nonprocedural language, although it contains a significant number of procedural commands. In the latter respect, it is structurally similar to BASIC, though the COMMAND names are different. In METAFILE, the command WRITE 5*45 will yield 225, just as in BASIC; PRINT 5*45 will do the same.

METAFILE procedures (programs) are identified by the file type (extension) PRO, just as BASIC programs are identified by the extension BAS, and as in interpreted BASIC, they can be run from the editor. All METAFILE commands must be entered in upper case, which given the IBM's lack of a CAPS LOCK indicator is somewhat of a nuisance.

An appreciation of METAFILE's capability can only be obtained by working with it, but it seems reasonable at least to list the classes of commands available and their functions (see Table II, page 83). Brief explanatory comments are noted in some cases.

Variables can be named and multidimensional subscripting is allowed as long as the total length of a data element is less than 235 characters. Variables are initialized by the command SET (with multiple initialization on one line allowed) and undefined with the command UNSET. Output is routed to SCREEN, PRINTER, or TEXT by use of the DEST command; default is SCREEN.

It seems useful to list the memory-dependent capacities of METAFILE as given in the Guide to Operations, since system performance is dependent on it. See Table III (page 83).

Summary

METAFILE is an excellent data base manager (record management, in METAFILE's lexicon) that utilizes a relational format. Each data record file has a data description specifying the attributes of the file and the items in its records. These are submitted in response to a command called DESCRIBE. Each item is capable of quite extensive description, including a comment line. Five files may be activated simultaneously.

Records may be updated, descriptions changed, new records inserted, and old ones deleted. Changes may be permanent or temporary, and the files may be searched and sorted with flexible querying commands. Fast lookup is implemented by a KEYING command. Columnar reports are produced with a LIST command and a row format with a ROW command. ITEMIZE will produce qualified records having non-null values.

The FORMAT commands allow extensive text management and preparation facilities, and flexible form and report generation is quite accessible. The ability to write METAFILE procedures to incorporate user-defined command sequences makes it possible to produce a combination of menus and procedures to permit system use by relatively inexperienced operators.

METAFILE goes well beyond data base management and word processing. The support from its programming language and the tight integration of the system offer the user enormous flexibility and suggest many interesting uses beyond those usually encompassed within those terms. If METAFILE gains acceptance in the user community, I would expect to see significant offerings of applications programs utilizing its facilities.

I am sure we have all heard the statement from high-level management: "I want to be able to press a button and see the status of XYZ." This is literally realizable with METAFILE. It is very easy to envisage writing a procedure with minimal prompts (whose implementation METAFILE makes easy) to load the appropriate file(s) and display the appropriate variables. Alternatively, the display could be done from the editor, exhibiting data variables within { } whose evaluation is

PRINT CONTROL COMMANDS – 14

SESSION FACILITIES – 8

(start, stop, HELP, print, etc.)

PROCEDURAL CONTROL

Clauses	8 commands
Commands	9 commands

FORMULA FACILITIES

Calculations	6 commands
Conjunctions	2 commands
Formatting	7 commands
Character Manipulation	5 commands
Built-In Functions	29 commands

All of the above are executable from a METAFILE procedure (program) or, as appropriate, from a command line or within the editor.

TEXT PREPARATION FACILITIES – 30

(in the EDIT mode only)

FILE COMMANDS

Record Files	9 commands
Text Files	6 commands
All Files	4 commands (erase, rename, transfer, connect)

RECORD COMMANDS

Update	8 commands
Qualification	3 commands (criteria)
Navigation	7 commands (moving within files)

TEXT LINE COMMANDS

Update	4 commands
Navigation	8 commands (locations)

DATA ELEMENT COMMANDS – 3

FRAME COMMANDS – 9

(these relate to the generation of forms and reports, including user interaction)

REPORTING COMMANDS – 18

Table 2.

METAFILE Commands

Total Memory[1]	Text Area	Separate Text/Sort	Sort Area[2]	Variables[3]	Items[4]
126K	26K	no[5]	26K	2175	3859
192K	63K	no[5]	63K	2943	4352
256K	63K	yes	63K	2943	4352
>256K[6]	63K	yes	63K	4100	6400

[1] Total memory means that memory assigned to METAFILE and does not include any assigned to a RAM disk.

[2] ((KeyLength)+3) * 10 (#Records to be sorted) must be less than 10 times the sort area (in bytes).

[3] Total number of characters in variables' names, values, and control information must not exceed value in table.

[4] Within active data file descriptions, the length of all items' names plus 10 characters per name must not exceed number in the table.

[5] These systems share the same area for text and sort. If you edit a text then sort a data file, text just edited is cleared from memory and must be recalled for re-edit.

[6] >256K means 320K to 640K in steps of 64K.

Table 3.

METAFILE Memory-Dependent Capacities

accessed by hitting the Evaluation key. METAFILE allows scrolling (using the Tab key) through the variable sequence. The existence of a macro facility would enhance this even further.

LOOKASIDE makes a user's "calendar" readily available for viewing or for updating without leaving the current work session. The enterprising programmer will, I am reasonably sure, come up with additional applications. The only aspect really lacking is a graphics capability; the lack of this capability is certainly not an oversight since METAFILE is records and text oriented, not number oriented, though it does more than a creditable job in that area.

The integration achieved in METAFILE and the ease both of utilizing the facilities in the editor and of transferring data, accompanied by the capabilities of the ASSISTANT and the LOOKASIDE facility, certainly make this a strong competitor in the integrated program sweepstakes. All this without the need of a hard disk, a mouse, or an enormous memory capability.

METAFILE's manners are impeccable. The prompts are to the point and understandable. Its error handling facilities are clean and clear as to correction. As noted, the only hang up that occurred was when basic operational instructions were not faithfully followed. The only real quirk was that relating to the disappearance of the DOS cursor after leaving a session.

The more I have used METAFILE in the process of preparing this review, the more I have been impressed both by its capabilities and by its depth. Alas, I have also been more and more depressed by the poor documentation and the failure of the implementation to utilize the IBM PC screen and keyboard features and to respond to its weaknesses (here I am referring to the lack of any flags for CAPS LOCK or NUM LOCK status and the concomitant requirement of COMMAND entry in upper case).

By using the term "depth," I imply that the user will continually find the program opening up new opportunities for application and use. When I initially ascertained the price of the package and of the major updates, I was somewhat taken aback. After exploring its potential, and considering the price of other DBM systems currently in the market place, I am not as taken aback as I was. I would expect, however, that a program as costly as this would not be as lax in dealing with details of screen and keyboard usage nor exhibit the deficiency of printer support. The question of performance speed in the crucial area of data base application is just not answerable in the context of a single review.

This is not, in my opinion, a program for the novice, publicity claims notwithstanding. It may be true that the novice could do a few trivial things right off, so to speak, but to really utilize the capabilities of this program, users must take the time and effort to learn something about it. I believe they will be rewarded for that effort. To be able to effectively utilize the program's resources will require some programming skills, again claims to the contrary notwithstanding.

What do I conclude? If the new releases contain a significant improvement in documentation, more responsiveness to the IBM's screen and keyboard features (with due respect for strengths and weaknesses therein), and enhanced printer support, the METAFILE must be seriously considered by those requiring an effective data base and management support system.

DDJ

MATH
SUBROUTINE
LIBRARY

Now, a library of **Numerical Methods Subroutines** for use with your **FORTRAN** programs.

Over sixty subroutines of:

FUNCTIONS	INTERPOLATION
INTEGRATION	LINEAR SYSTEMS
MATRICES	POLYNOMIALS
NON-LINEAR SYSTEMS	DIFFERENTIAL EQ

Versions available for several FORTRAN compilers running under **CP/M-80** and **MS-DOS (PC-DOS)**.

Cost, $250.
Manual available, $25.

microSUB:MATH

foehn consulting, PO Box 5123, Klamath Falls, OR 97601 503/884-3023

CP/M and MS-DOS are trademarks of Digital Research Corp and MicroSoft Corp respectively

Circle **no. 21** on reader service card.

Elegance

Power

Speed

C Users' Group
Supporting All C Users
Box 287
Yates Center, KS 66783

Circle **no. 13** on reader service card.

SWIG©

SOFTWARE WRITERS INTERNATIONAL GUILD

SCHEDULED SWIG ACTIVITIES & MEMBERSHIP BENEFITS

(1) $10,000 PROGRAMMING CONTEST (Members only)

(2) NATIONAL COMPUTER WEEK (May 11-May 20, 1984)

(3) ANNUAL CONFERENCE AND SOFTWARE AWARDS CEREMONY (During National Computer Week)

(4) CONSULTANT REGISTRY (With computer store referral system for customized software)

(5) JOB PLACEMENT SERVICE (Free to individual members, fixed maximum fee to companies)

(6) FREE SEMINARS & MEETINGS LOCALLY

(7) SOFTWARE LIBRARY LENDING & EXCHANGE SERVICE (Professional quality assemblers, utilities, games, etc.)

(8) SOFTWARE LOCATION SERVICE (For companies & individuals-if it exists, **SWIG** will find it. If not, see #9)

(9) SOFTWARE DEVELOPMENT SERVICE (From novice to scientist, **SWIG** members can work on any project-from applications to games to R&D)

(10) LEGAL SERVICE

(11) AGENT (**SWIG** can represent you in sales to software publishers)

(12) 24 HOUR - 7 DAY BULLETIN BOARD SYSTEM (BBS) ACCESSIBLE BY COMPUTER FREE

(13) AND MORE!!!!

THE LARGEST PAID MEMBERSHIP PROGRAMMERS GUILD - OVER 5,000 MEMBERS WORLDWIDE!!

MEMBERSHIP APPLICATION FOR **SOFTWARE WRITERS INTERNATIONAL GUILD**

NAME ______________________________

ADDRESS ______________________________

CITY ____________ STATE ________ ZIP ________

PHONE # (   ) ____________

• CLASSIFICATION:

☐ NOVICE ☐ BEGINNER TO ADVANCED

☐ ADVANCED WITH ON THE JOB EXPERIENCE ☐ RESEARCH/SCIENTIST

• WHAT EQUIPMENT DO YOU HAVE EXPERIENCE WITH &/OR ACCESS TO &/OR PLAN TO BUY?

☐ MAINFRAME ☐ MINI ☐ MICRO ☐ DESIGN/R&D

BRAND NAME(S): ☐ IBM ☐ XEROX ☐ APPLE ☐ TI

☐ COMMODORE ☐ RADIO SHACK ☐ ATARI ☐ OSBORNE

☐ TIMEX/SINCLAIR ☐ NORTH STAR ☐ HEWLETT PACKARD

☐ OTHER ______________________________

• AREAS OF INTEREST:

☐ DATA PROCESSING ☐ BUSINESS APPLICATIONS ☐ GRAPHICS

☐ LEGAL ☐ VOICE ☐ MEDICAL ☐ APPLIANCE (HOME) CONTROL

☐ ROBOTICS ☐ GAMES ☐ MUSIC ☐ R&D ☐ OTHER ______

• MEMBERSHIP ACTIVITIES AND SERVICES OF INTEREST:

READ THE LIST ON THE LEFT AND CIRCLE THE NUMBERS BELOW THAT APPLY.

1 2 3 4 5 6 7 8 9 10 11 12

☐ I HAVE ENCLOSED $20 ANNUAL MEMBERSHIP FEE ☐ CK ☐ MO
(MAKE CHECK PAYABLE TO: **SWIG**)

RETURN TO: **SWIG**
P.O. BOX 87
STONY POINT, NEW YORK 10980
(914) 354-5585

SWIG© SOFTWARE WRITERS INTERNATIONAL GUILD

Circle **no. 58** on reader service card.

16-BIT SOFTWARE TOOLBOX

by Ray Duncan

More on MS-DOS EXEC Function

Morton F. Kaplon, of Pomona, NY, writes: "I read with some interest your article in the December 1983 issue of *DDJ* . . . . I certainly do agree with you that the documentation on the DOS function 4BH is less than transparent, though my prize for opacity in DOS is still easily won by Chapter 13, 'Using Extended Screen and Keyboard Control.'

"In any event, my approach was somewhat different than yours and the program I had created offers a somewhat more extensive demonstration of the utility of that call. It allows the user to enter from the program anything which can be entered from the command line, albeit in two steps if there is a tail to the command line.

"Given this program, I believe it is fairly easy to see how one can construct a 'DOS MANAGER' for the user who wants to be prompted through DOS." Morton was kind enough to provide his program on a diskette, and we are printing a slightly modified version of it in Listing One (page 88).

PC-DOS Manual Typo

PC-DOS 2.0's manual contains a small typographical error in Chapter 13 which may confuse programmers who are attempting to use the Extended Screen Control. The command for "Erase in Line" (usually known as Clear to End of Line) is "ESC [K", not "ESC [k" as given in the text.

IBM PC Junior

We have now received a PC Junior to play with and are making many interesting discoveries. The graphics support on the Junior is considerably enhanced over the standard IBM PC, with programmable palettes and the ability to display sixteen colors in medium-resolution graphics mode or four colors in high-resolution mode.

One issue that has been completely ignored in the popular magazines is the raw performance of the Junior. Apparently, everyone assumes that because it uses a 8088 microprocessor at 4.77 MHz, it will execute programs at a speed comparable to a normal PC. Well, friends, it just ain't so. just ain't so.

The PC Junior does not have dual ported memory for the video refresh buffer. Consequently, the video controller must "steal cycles" from the CPU while it is updating the screen. It turns out that two out of three memory cycles for the main machine RAM are allocated to the 6845 controller, and only one out of three for the 8088. Running several standard benchmarks, I found that the execution speed of a computation-bound program on the Junior is exactly 45% of its speed on a normal PC. Programs which perform a great deal of screen I/O appear to suffer even more, though I haven't measured this exactly yet.

The literature on the Junior leads one to believe that programs executing out of a ROM cartridge will execute much faster, since the video controller will not be accessing that memory. I'll check this out and provide more information next month.

The performance of PC-DOS 2.1, which is provided with the Junior, has also been degraded. Apparently the disk drives provided on the Junior have much longer seek and head settle times, so the BIOS disk driver's delay factors have been lengthened accordingly. Access to files by a disk-based program is painfully slow – just the process of opening a file seems to take over a second.

MS-DOS Volume Labels

I had surmised from reading the MS-DOS 2.0 manual that I could change the volume label on a disk by simply performing a search with an extended file-control-block for the first directory entry with a "volume" attribute byte, requesting a "delete" on any filename found in that manner, then requesting a "create" function with the new name. However, this path is fraught with peril. The delete appears to work OK and returns a successful status code, but it clobbers the file allocation table unmercifully (see Figures 1 and 2, page 87).

I was unable to figure out exactly what was going on here. Examining a dump of the FAT and directory sectors, it is readily seen that a volume label is simply a normal appearing directory entry with the attribute byte set to 8, date and time fields filled in, and the "starting cluster" and "file size" fields zeroed. Except for the attribute byte, this is just the same as a directory entry for a file that was created but never had any data written into it; however, a delete of such an empty file works fine. Evidently, finding the volume attribute bit turned on during a delete function sidetracks PC-DOS into some nutty logic or other.

By experimentation, I found that the following sequence will safely add or change a volume label under PC-DOS or MS-DOS 2.0:

1. Set the disk transfer address to a scratch buffer 128 bytes long.

2. Using an extended file control block with the format shown in Listing Two (page 92), perform a "search for first" function (#17). If the register AL is returned as 0FFH, then the disk has no label – go to step 5.

3. If function 17 returned register AL = zero, the buffer contains a simulated extended file control block with the volume name in bytes 8-19. Move your new volume name (11 characters) to buffer + 24.

4. Passing the address of the scratch buffer in registers DS:DX, request a "rename" function (#23). If AL returns 00, the volume name was successfully modified and you are finished, otherwise something is horribly wrong with your system.

5. (Come here if the disk has no previous label.) Replace the wild-card characters shown in the extended file control block with the desired 11-character volume name. Passing the address of the extended fcb in DS:DX, request a PC-DOS "create" function (#22). AL returned as 0 implies success; AL returned as 0FFH means failure (usually due to a full directory).

I have provided an extract of the final working program as Listing Two (page 92) to get you started. Interestingly, this procedure *always* adds the volume label to the root directory no matter what your "current" subdirectory is.

I made another surprising discovery. Even though volume names were added in MS-DOS 2.0, the new version 2.0 extended file and record handling functions don't support adding or changing the volume labels. When I first tried this, I thought that I could probably use the new

CREAT call (3CH) and then use function 43H (CHMOD) to set the attribute on the new file to "volume"; however, the CHMOD call always returns an "access denied"error code.

Finding Size of TPA under MS-DOS 2.0

On the IBM PC, the Wang Professional Computer and presumably most other MS-DOS machines, there is a special BIOS call or some other method to get the amount of machine memory. Although it would be nice if RAM size could be obtained in a machine-independent way, there is no obvious MS-DOS function call that will do the trick. However, it can be done, although circuitously!

The method hinges on the "Modify Memory Allocation" call, MS-DOS function 4AH. It is defined as follows in the MS-DOS manual: "on entry, ES contains the segment of the block, BX contains the new requested block size in paragraphs. DOS will attempt to 'shrink' or 'grow' the specified block. If the call fails on a grow request, then on return BX contains the maximum block size possible . . . . "

Knowing that part of the MS-DOS procedure for loading an application program includes allocating a memory block to it, we can simply request an increase of our program's area to the maximum possible, inspect the amount of RAM that is actually given to us, and then add that to the segment address our program is based at in order to find the actual size of machine memory. See Listing Three (page 93) for an example subroutine that returns RAM size in Kbytes.

CP/M Discrepancies

While porting a program that performed ramdom file I/O from CP/M-86 to CP/M-68K, I discovered that they are not quite symmetric with respect to file control block structure. Specifically, bytes 31-34 (r0, r1, r2) have different meanings under the two operating systems (see Figure 3, at right). This was obviously done so that the lower 16 bits of the random record number would fall on a word boundary for the 68000, since 16-bit accesses on off addresses are verboten. However, it would have been nice if DRI had printed a warning about this in the manual somewhere.

DDJ

(Listings begin on page 88)

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 196.**

```
Drive:B    Sector:1        Cluster:FAT#1     Dir:ROOT.DIR

            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

000000:    FC FF FF 03 40 00 05 60 00 07 80 00 09 A0 00 0B
000010:    C0 00 0D E0 00 0F 00 01 11 20 01 13 40 01 15 60
000020:    01 17 80 01 19 A0 01 1B C0 01 1D E0 01 1F 00 02
000030:    21 20 02 23 40 02 25 60 02 27 80 02 29 A0 02 2B
000040:    F0 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000050:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000060:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000070:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00


Drive:B    Sector:5        Cluster:     Dir:ROOT.DIR

            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

000000:    57 4F 52 4B 20 20 20 20 20 20 20 08 00 00 00 00
000010:    00 00 00 00 00 00 6F 14 32 08 00 00 00 00 00 00
000020:    46 4F 52 54 48 20 20 20 43 4F 4D 20 00 00 00 00
000030:    00 00 00 00 00 00 D9 08 6F 07 02 00 04 53 00 00
000040:    00 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6
000050:    F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6
000060:    00 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6
000070:    F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6
```

Figure 1.

First block of File Allocation Table and Disk Directory before Delete performed on volume label using extended file control block. This disk contains only two directory entries and one file.

```
Drive:B    Sector:1        Cluster:FAT#1     Dir:ROOT.DIR

            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

000000:    FC FF FF 03 40 00 05 60 00 07 80 00 00 00 00 00
000010:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000020:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000030:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000040:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000050:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000060:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000070:    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 2.

First block of File Allocation Table after Delete function call performed on volume label. Note that many of the clusters allocated to the normal file on the disk are now missing.

Byte name	Offset	CP/M-86	CP/M-68K
r0	33	1sb*	msb+
r1	34	2sb	2sb
r2	35	msb	1sb

*1sb = least significant byte
+msb = most significant byte

Figure 3.

Discrepancy in file control block structure between CP/M-86 and CP/M-68K.

16-Bit Toolbox (Text begins on page 86)

Listing One

```
        name     docom
        title    'DOCOM - Load and execute programs'
        page     55,132
;
; DO-COM.ASM --- Load and execute COM programs
; demonstrates use of PC-DOS function 4BH
; Morton F. Kaplan, 11 White Birch Drive, Pomona, NY 10970
;
; Definition of Function 4BH call:
; DS:DX points to ASCIIZ filespec
; ES:BX points to parameter block for load
; AL = function type:
;       0 is load and execute. Program must save registers
;         and must restore SS and SP.
;       3 is load only (may be used for overlays)
; Parameter Block
; AL= 0: Word segment address of environment string to be passed,0 value
;            for address will cause parent's environment to be inherited
;        DWORD pointer to command line to be placed at PSP+80h
;        DWORD pointer to default FCB to be passed at PSP+5ch
;        DWORD pointer to default FCB to be passed at PSP+6ch
;
; AL= 3: WORD segment address where file will be loaded
;       WORD relocation factor to be applied to the image.
;

; In this program, the filespec has to contain the filename and extension,
; a path may also be included.  Thus, to load and execute the program
; testmac.com on b: from drive a:, do as follows:
;
; A>docom
; Enter filespec to be loaded and run          (this prompt appears on screen)
; >b:testmac.com<enter>
; Enter command tail, if none then <enter>     (this prompt appears on screen)
; >
; Program then executes.


;
; Build this program from file DOCOM.ASM as follows:
; MASM DOCOM,DOCOM,DOCOM,DOCOM
; LINK DOCOM,DOCOM,,,
; EXE2BIN DOCOM.EXE DOCOM.COM
; ERASE DOCOM.EXE
;
;
; define necessary MACROs

dos     macro   fn                          ;fn is dos function to be called
        mov     ah,fn                       ;fn passed to ah
        int     21h                         ;dos interrupt
        endm                                ;end of macro

kb_in   macro                               ;chr typed at KB returned in al
        dos     1                           ;function 1
        endm
```

```
crlf    macro
        mov     dl,10                   ;line feed
        dos     2
        mov     dl,13                   ;carriage return
        dos     2
        endm

cue     macro                           ;perform CR-LF sequence
        crlf                            ;then display CUE mark
        mov     dl,'>'
        dos     2
        endm

bfr_kbin macro  bfr_addr                ;buffered keyboard input
        mov     dx,offset bfr_addr      ;dx points to input buffer
        dos     0ah                     ;dos function call
        endm                            ;end of macro

print_string macro bfr_locn             ;print string at bfr_locn
        mov     dx,offset bfr_locn      ;point dx
        dos     9                       ;function call 9 of dos
        endm                            ;end of macro

; END of Macro definitions

        codeseg segment para
```

(Continued on next page)

C Programmers: Program three times faster with Instant-C™

Instant-C™ makes programming three or more times faster by eliminating the time wasted by traditional compilers. Many repetitive programming tasks are automated to make programming less frustrating and tedious.

- Two seconds elapsed time from completion of editing to execution.
- Full-screen editor integrated with compiler; compile errors set cursor to trouble spot.
- Editor available any time during session.
- Symbolic debugging; single step by statement.
- Automatic recompilation when needed. Never a mismatch between source and object code.
- Directly generates .COM, .EXE, or .CMD files.
- Follows K & R—works with existing source.
- Single, integrated package.
- Works under PC-DOS*, MS-DOS*, CP/M-86*.

More productivity, less frustration, better programs. **Instant-C™** is $500. Call or write for more information.

Rational Systems, Inc.
(617) 653-6194
P.O. Box 480
Natick, Mass. 01760

*[Trademarks: PC-DOS (IBM), MS-DOS (Microsoft), CP/M-86 (Digital Research, Inc.) Instant-C (Rational/Systems, Inc.)]

Circle **no. 50** on reader service card.

A Professional Quality Z80/8080 Disassembler

REVAS Version 3

Uses either ZILOG or 8080 mnemonics
Includes UNDOCUMENTED Z80 opcodes
Handles both BYTE (DB) & WORD (DW) data
Disassembles object code up to 64k long!
Lets you insert COMMENTS in the disassembly!

A powerful command set gives you:

INTERACTIVE disassembly
Command Strings & Macros
On-line Help
Calculations in ANY Number Base!
Flexible file and I/O control
All the functions of REVAS V2.5

REVAS:

Is fully supported with low cost user updates
Runs in a Z80 CPU under CP/M*
Is normally supplied on SSSD 8" diskette
Revas V 3...$90.00 Manual only...$15.00
California Residents add 6½% sales tax

REVASCO
6032 Charlton Ave., Los Angeles, CA. 90056
(213) 649-3575

*CP/M is a Trademark of Digital Resaerch, Inc.

Circle **no. 51** on reader service card.

16-Bit Toolbox (Listing Continued, text begins on page 86)

Listing One

```
        assume  cs:codeseg,ds:codeseg,es:codeseg

        org     100h                    ;required for COM

start:  jmp     docom                   ;jump over data section


error1      db  'Memory Shrink Failure','$'
error2      db  'Error in Loading Program','$'
prompt_Ø    db  'Enter filespec to be loaded and run ','$'
prompt_1    db  'Enter command tail, if none then <enter> ','$'

stk_ptr     dw  Ø                       ;save contents of SP
stk_seg     dw  Ø                       ;save contents of SS

                                        ;PARAMETER BLOCK
                                        ;
parm_blk dw     Ø                       ;segment address of environment
                                        ;
                                        ;string passed,use parent
                                        ;pointer to command line at
                dw      offset psp8Ø+1  ; at psp+8Øh, +1 needed
                dw      ?               ;to allow for structure of
                                        ;psp8Ø using buffered
                                        ;input from keyboard
                                        ;
                dw      offset fcb1     ;pointer to default FCB1 at psp+5ch
                dw      ?
                dw      offset fcb2     ;pointer to default FCB2 at psp+6ch
                dw      ?


                                        ;DATA FOR PARAMETER BLOCK
                                        ;
                                        ;buffer of 2Ø char for buffered
                                        ;line input, can be increased
 filespec       db      2Ø,?,2Ø dup (?)
 psp8Ø          db      32,?,32 dup (?) ;32 chars in buffer,increasable
 fcb1           db      1               ;drive id
                db      11 dup ('?')    ;filename and extension
                db      25 dup (Ø)      ;rest of fcb
 fcb2           db      1               ;drive id
                db      11 dup ('?')    ;filename and extension
                db      25 dup(Ø)       ;rest of fcb


 docom:                                 ;start of effective code
                push    cs              ;initialize DS and ES
                pop     ds
                push    cs
                pop     es              ;initialization finished
                mov     bx,5Ø           ;save 5Ø*16 bytes for program
                dos     Ø4ah            ;es already set to code segment
```

```
               crlf
               jc      not_set           ;check for memory shrink
               print_string prompt_Ø     ;display prompt
               cue
               jmp     over_erl
not_set:                                 ;display error message
               print_string errorl
               mov     dx,ax             ;print error code in ax
               add     dl,48             ;convert to ascii
               dos     2                 ;display on kb
               jmp     quit

over_erl:
               bfr_kbin filespec         ;get filespec
               crlf
               sub     ax,ax             ;clear to Ø
               mov     al,filespec+1     ;get length of string input
               mov     di,offset filespec+2
               add     di,ax             ;to end of string
               mov     byte ptr [di],Ø   ;terminate with Ø

               print_string  prompt_1    ;prompt for command tail
               cue
               bfr_kbin psp8Ø            ;get command tail
               crlf
               sub     ax,ax             ;clear to Ø
```

(Continued on next page)

DESMET C $109

PCDOS+ — CP/M-86++ — MP/M-86++ — CCP/M-86++

- **C DEVELOPMENT PACKAGE**
  C Compiler, Assembler, Linker, and Librarian
  Full Screen Editor — Native 2.0 Support
- **COMPLETE IMPLEMENTATION**
  Everything in **K & R** (incl. **STDIO**)
  Small case — 64K code, 64K data/stack
  All 8088/8087 data types
  Intel assembler mnemonics
- **OUTSTANDING PRICE/PERFORMANCE**
  Aug. '83 *BYTE* "SIEVE" Benchmark
  135 bytes compiled — 6144 bytes linked
  65 sec. compile (fdisk) — 11.5 sec. run (10 iter)
- **SUPPORT LIBRARIES**
  Both 8087 **and** Software Floating Point
  PC Keyboard/Display
  Trig/Log functions
- **UPDATES**
  Updates and Bugs reported in newsletter
  No time/number limits

+Trademark IBM ++Trademark Digital Research

ORDER FORM

Name ______________________

Address ______________________

____

____

Phone ______________________

OS: ☐ PCDOS ☐ MSDOS ☐ CP/M-86, MP/M-86, CCP/M-86
DISKS: ☐ 8"(3740) ☐ 5.25" SS (160KB) ☐ 5.25" DS (320KB)
MACHINE? ______________ (IBM, ZENITH, NEC, etc.)

DeSmet C Package	$109	______
Manual Only (refunded with order)	$20	______
Update Only	$20	______
	Sub Total	______
California Residents add 6 % Sales Tax		______
	Total	______

Send To: C WARE
P.O. Box 710097
San Jose, CA 95171-0097
(408) 736-6905

Circle **no. 14** on reader service card.

16-Bit Toolbox (Listing continued, text begins on page 86)

Listing One

```
                mov     al,psp80+1          ;get length of string input
                mov     di,offset psp80+2
                add     di,ax               ;to end of string
                                            ;terminate with carriage return
                                            ;and null byte
                mov     byte ptr [di],13
                mov     byte ptr [di+1],0
                mov     cs:stk_ptr,sp       ;save SS and SP in
                mov     cs:stk_seg,ss       ;variables addressable by CS
                mov     di,offset parm_blk
                mov     [di+4],es           ;seg address to psp80
                mov     [di+8],es           ;seg address to fcb1
                mov     [di+12],es          ;seg address to fcb2
                                            ;point to parameter block
                mov     bx,offset parm_blk
                                            ;point to ASCIIZ filename
                mov     dx,offset filespec+2
                mov     al,0                ;load and execute
                dos     4bh                 ;dos function to execute program
                mov     ss,cs:stk_seg       ;restore SS:SP pair
                mov     sp,cs:stk_ptr
                jnc     quit                ;jump if no load error
                print_string error2         ;print error message
quit:           dos     4ch                 ;back to dos,
                                            ;easiest way to terminate, can also
                                            ;pass code back with this call

codeseg         ends
                end     start
```

End Listing One

Listing Two

```
          .
          .
          .
          mov  dx,offset buffer
          mov  ah,26            ;set disk transfer address
          int  21h              ;to scratch area (DS:DX) for
                                ;dictionary search.
          mov  dx,offset xfcb   ;DS:DX=addr extended fcb
          mov  ah,17            ;"search for first match"
          int  21h
          cmp  al,0ffh          ;successful?
          je   no_label         ;no label on disk, jump.
          jmp  label_found      ;label found, jump.
          .
          .
          .
xfcb      db   0ffh             ;flag signifying extended fcb
          db   5 dup (0)        ;reserved (should be zeroes)
          db   8                ;volume attribute byte
          db   0                ;drive code (set by program)
          db   11 dup ('?')     ;wild card filename & ext.
          db   25 dup (0)       ;remainder of fcb (not used)

buffer    db   128 dup (?)      ;buffer for directory search
```

End Listing Two

Listing Three

```
;
; Return memory size (Kbytes) in AX.
; Calling arguments:  none.
: Destroys AX, BX, CX.
; This routine assumes that ES is pointing
; to the Program Segment Prefix (true
; on original entry from MSDOS for either
; an EXE or a COM type program)
;
ramsize    proc      near
           mov       bx,-1       ;request 65535 paragraphs!
           mov       ah,4ah      ;MSDOS modify block function
           int       21h         ;returns paragraphs available
                                 ;in register BX
           mov       ax,cs       ;add that to the segment of
           add       ax,bx       ;the base of our program
           mov       cx,6        ;then shift paragraphs right
           shr       ax,cx       ;six places to get Kbytes
           ret                   ;back to caller
ramsize    endp
;
```

End Listing

WORKMAN & ASSOCIATES
USER-FRIENDLY SOFTWARE

WRITE

The Writer's Really Incredible Text Editor lives up to its name! It's designed for creative and report writing and carefully protects your text. Includes many features missing from WordStar, such as sorted directory listings, fast scrolling, and trial printing to the screen. All editing commands are single-letter and easily changed. Detailed manual included. WRITE is $239.00.

WORKMAN & ASSOCIATES
112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

All US orders are postpaid. We ship from stock on many formats, including: 8", Apple, Osborne, KayPro, Otrona, Epson, Morrow, Lobo, Zenith, Xerox. Please request our new catalog. We welcome COD orders.

Circle no. **68** on reader service card.

C BUILDING BLOCKS

Save a year of development.

□ **IBM PC* Building Blocks**	$149
Video & IO routines, string functions, asynchronous port control, date, time, random numbers, printer control. Over 200 functions.	
□ **Mathematics Library**	$99
Logarithms, trigonometric functions, square root, random numbers.	
□ **Communications Library**	$149
For Hayes Smartmodem†, modem 7 and xmodem.	
□ **B-trees & Virtual Memory Management Library** ...	$149
List handling features.	
□ **Advanced Building Blocks**	$ 99
Julian dates, dBaseII access, data compression, text windows.	

All Building Blocks use the 'IBM PC Building Blocks'.
Credit Cards Accepted. Single User License.
Multiuser licenses available. (Mass. add 5%)

NOVUM ORGANUM

29 Egerton Road, Arlington, MA 02174   Tel: (617) 641-1650

(TM) IBM*   (TM) Hayes Microcomputer Products†   (TM) Ashton-Tate*

Circle no. **40** on reader service card.

C/UNIX PROGRAMMER'S NOTEBOOK

by Anthony Skjellum

This month's Notebook includes reader response to my December 1983 column, where I discussed perceived deficiencies in the Unix operating system, as well as a follow-up discussion on runtime libraries and link format incompatibilities.

It was also my intention to discuss two C compilers that support the large memory concept on the 8086. But since I have yet to receive one of the compilers, this comparative review will have to be included in a future column.

Several more letters have come in concerning C layout standards. While we don't have room to include their comments at present, they will undoubtedly lead to several columns on related topics of interest.

Comments about Unix

In the December column, I mentioned several aspects of Unix that I believe constitute weaknesses. Tony LiCausi of Canoga Park, California, sent me a letter with his dissenting opinions. He begins by remarking:

"Unix was designed for the programming environment, for use by folk already familiar with computers. Unnecessary I/O slows down a program's performance and hence should be eliminated. Thankfully, verbosity is often eliminated in Unix as the default option. However, many of the programs do have command line switches to increase the progress reporting of the program."

Unix was designed to circumvent the flaws of existing operating systems. Under Unix, modular programs are coupled via pipelines to create a convenient working environment and to reduce repetition of effort. Furthermore, Unix offers unequalled portability between large and small computer systems. Since it is successful in these respects, Unix's popularity is growing; hence, more and more computer users are gaining access to machines that use Unix. Many of these users are familiar with computers and programming, while others are novices to both computing and Unix.

Let's assume that a user is well versed in programming but is new to Unix; according to Mr. LiCausi, this user belongs to the group of people for whom Unix was designed. I'll assume that the user has an application in mind, since the average user does not write programs as ends unto themselves. To succeed in implementing an application, the user must learn about various aspects of Unix, including various shell commands, system calls, and runtime libraries.

I submit that this type of user will be slowed down significantly by the overall poor quality of the Unix documentation. Since examples are typically both terse and complicated, the typical user could spend an enormous amount of time circumventing a single snag. Imagine linking the mathematical function library as part of a C compilation. Two seemingly possible ways for doing this are:

```
cc -o test test.c -lm
        or
cc -lm -o test test.c
```

It turns out that the former method is correct, while the latter produces undefined function error messages at the linker stage. However, the documentation does not indicate that the "-lm" must be near the end of the line.

Our hypothetical user will also be slowed down by the cryptic nature of the error messages. True, the more sophisticated users require little or no prompting, and only gentle nudges when they make an incorrect entry, but the vast majority of users probably don't belong to this category.

Finally, the user will be slowed down by inexplicable and undocumented bugs in systems such as **troff** and **eqn**, which I mentioned before. Furthermore, circumventing bugs requires highly specific knowledge, and the solutions, even if available, may not be widely known. On the Unix machine I use, only a small group of people is aware of the procedure for avoiding some common but annoying bugs in **eqn**.

Mr. LiCausi makes a very valid point concerning this:

"Software bugs are not acceptable in any form at any time. Unix from Bell Labs is sold as unsupported software. Their attitude is, 'OK, we'll sell it to you, but we don't want to hear from you.' There is no mechanism to report or correct bugs at the source. Source code is not typically available to the average user. Source code is guarded on Unix like gold, just like any other commercial software. Having done debugging of the f77 compiler on one system, I found access to other software (for examples and integration) severely limited."

Thus, while Mr. LiCausi contends that Unix was designed for people with programming experience, he sets his standard for "unnecessary I/O" and "program performance" based on the most sophisticated level of Unix user. Furthermore, it is still not clear to me why the sophisticated users should be more willing to stand for poor quality documentation and user-unfriendly software than everyday users. Finally, it is not at all apparent to me from practical experience that "many" of the standard shell commands support extended progress reporting.

Mr. LiCausi goes on to state:

"The secret society of Unix is partially the result of the paranoia of 'unauthorized access.' The remainder is due to the lack of commercial incentive. The incentive has obviously increased, as can be evidenced by the proliferation of books and articles on Unix and C."

I stated in my previous column that some users perceive Unix as a secret society because they cannot grasp certain aspects of its operation based on the written and on-line documentation alone. Only through initiated members can the knowledge (and undocumented tricks, patches, etc.) be obtained. However, other popular operating systems do not have the same aura as Unix. For example, CP/M users are not considered a "secret society" even though CP/M has nontrivial aspects. Nor do minicomputer operating systems such as VAX/VMS (DEC) and AOS/VS (Data General) have the mysterious quality of Unix. That's because they have much more documentation to explain the nitty-gritty of using the system.

Mr. LiCausi finishes by stating:

"Unix is not user-unfriendly. Admittedly it is beginner-hostile. Changes will be required to adopt Unix en masse (i.e., for the micro), but I hope that it does not spell the end of 'silent' software."

My final counter-remark is that most users never become sufficiently sophisticated in enough diverse aspects of Unix to escape the feeling of user-hostility. I personally still run into difficulties, even though I've used Unix for almost four years. As soon as I use aspects of Unix that are not part of my daily routine, "gottchas" appear.

Other Points of View

H. T. Gordon of Berkeley, California, sent in a letter concerning Unix. He writes:

"It was a pleasure to read the Skjellum critique of [Unix] in *DDJ* No. 86. Since C/Unix is becoming the sacred cow of the computer Brahmins, few have the courage to point out its flaws. It was designed by one of the elite, for elite users. To them it's concise and elegant, even 'obvious.' Only rarely is any explanation needed. If it is, a terse one will do. Prophets inspired by the high gods cannot waste their time spelling things out for mere mortals in words of one syllable."

I don't think that the state of C/Unix is as bad as Mr. Gordon suggests. While there are problems, I think that Unix has many strengths and these strengths will be the basis for future operating systems. It is the purpose of this column to examine strengths and weaknesses of C and Unix, and to propose solutions as well as to point out the problems. I encourage readers to pose solutions to these and other problems they encounter.

Low Level Input-Output in C

Patrick Cawood of Los Angeles writes:

"I read with great interest your last article on tendencies in Unix to produce poor operator interaction programs. I seem to have met some of the same in C.

"In order to provide a secure operator interface, one does not echo a keyboard character to the screen until it has been examined and approved by the program. But **getch()** function automatically echoes – even line feeds, up arrows, etc.! Function **putch()** provides an automatic line feed after printing on the screen!"

He goes on to state:

"I simply cannot believe that anyone would wittingly design these functions as they are, or not provide any alternative hardware interfaces. Especially considering some of the tasks I've heard were written in C. But perhaps these people writing serious software were all forced to write their own hardware interfaces."

The problem that Mr. Cawood is referring to exists in a number of non-Unix C compilers. To begin with, let's review the problem in the Unix environment. Under Unix, **putc()** and **getc()** acquire and return a single character, respectively. To offload the host system, however, many terminal interface boards programmatically handle the user input-output in lines. Thus, before any input is received by the host, a whole line must be entered. Naturally, the characters are echoed by the terminal interface hardware/firmware, and only limited line editing is permitted. On output, a whole line is buffered up before transmission to the terminal.

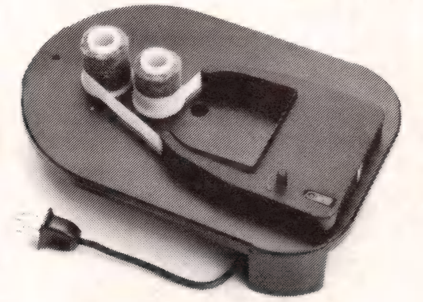
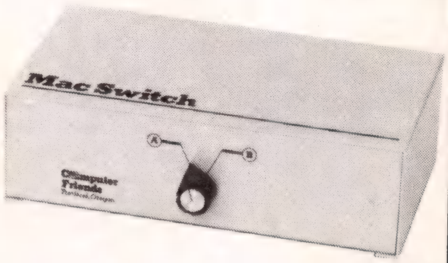
This process can be overcome by use of the "raw" terminal mode (raw implies no host character processing). In this mode, the program is completely responsible for input-output. This mode is much more expensive in terms of input-output cost, since the host must handle an inter-

Re-ink any fabric ribbon for less than 5¢. Extremely simple operation. We have a MAC INKER for any printer. Lubricant ink safe for dot matrix printheads. Multi-colored inks, uninked cartridges available. Ask for brochure. Thousands of satisfied customers.

$54.95 +

Mac Switch lets you share your computer with any two peripherals (serial or parallel). Ideal for word processors—never type an address twice. Ask us for brochure with tips on how to share two peripherals with MAC SWITCH. Total satisfaction or full refund.

$99.00

Order Toll Free 1-800-547-3303

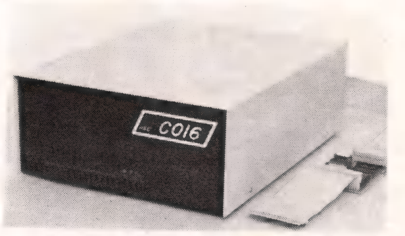
Computer Friends
6415 SW Canyon Court
Suite #10
Portland, Oregon 97225
(503) 297-2321

Mac Inker & MacSwitch

Circle no. 9 on reader service card.

SAVE YOUR 8 BIT SYSTEM
Join the 16/32 Bit Revolution Through Evolution, For Under $600

APPLE 2E™
RADIO SHACK TRS80™
ZENITH™
OSBORN™
ALTOS™
KAYPRO™
COLUMBIA™
TELEVIDEO™
DEC™
or OTHER Z80 BASED SYSTEM



Would you like to run CPM86™, CONCURRENT CPM86™, CPM68K™, MS-DOS™, IBM™ PC APPLICATIONS, DEVELOP and TEST 16/32 BIT SOFTWARE, or add an INTELLIGENT HIGH SPEED RAM DISK to YOUR SYSTEM?

You can with the HSC CO16 ATTACHED RESOURCE PROCESSOR:

CO16 with either 8086, 80186, or 68000 16 bit micro-processor and up to 768K Bytes of parity checked RAM may be connected to virtually any Z80 based computer system or APPLE 2E computer system.

An 8 bit system equipped with CO16 retains all of its original capabilities PLUS it has the added ability to run most 16 bit operating systems and applications.

Prices start at $595.00 which includes the CO16 processor with 128K bytes of memory, two volume user manual, and the HSC Software Development System. Hardware options include memory expansion to 768K bytes, attractive desk top enclosure with power supply, and the INTEL™ 8087 Math Co-processor (available on CO1686 only). Software options include CPM86, CPM68K, Concurrent CPM86, or MS-DOS™ operating systems. The UNIX™ operating system will be available in the second quarter of 1984.

For Information on CO16 Contact
HSC INC.
Box 86
Herkimer, NY 13350
(315) 866-2311

HSC INC.

Dealer and OEM inquiries are invited.

Circle no. 24 on reader service card.

rupt for each input character and perform an output request for each printed character. However, as I mentioned in the previous column, this is the only way for a program to get full control of what is entered and appears on the display device.

With the introduction of C to microcomputers, compiler implementors often based the behavior of the runtime libraries on their Unix experiences rather than on *The C Programming Language*. Whole lines are prebuffered by typical C runtime libraries before a single character is received by **getc()**; conversely, a whole line of output is internally buffered before it is printed on the display. Thus, the runtime libraries of microcomputer C compilers often emulate the terminal hardware found on real Unix systems.

However, this is not what **putc()** and **getc()** are supposed to do. In the truest sense, raw mode is the fundamental terminal mode. These functions should really work on a character-by-character basis. C libraries should permit selection between the "raw" and "cooked" modes (and echo and no-echo modes), thus permitting input-output flexibility without resorting to assembly language routines.

One C compiler that correctly handles low level input-output is the Q/C compiler from The Code Works. Single characters, for example, are required only when a **getchar()** call is made. While it doesn't seem possible to turn off the echo, this can be effected under Q/C since The Code Works is kind enough to provide runtime and compiler source code.

Link Formats

In the October 1983 column I discussed incompatibility between link formats. Readers had quite a bit to say about this and their comments are summarized here.

Guy Scharf of Mountain View, California, writes:

"I find the incompatibility of linkage editor formats to be a real problem. For example, I want to use Digital Research's Access Manager and Display Manager 86 with C. I would prefer CI-86 (because I am used to it) but have to switch to DRI's C because of the link format. Another compiler to learn and idiosyncrasies to surmount."

He concludes: "I'm not sure what to do about this problem (except complain)."

David D. Clark of State College, Pennsylvania, writes:

"The big problem isn't really the format of linkable files. Even compilers that use Microsoft's M80 and L80 will not allow linkage to code produced by different compilers. The function-calling protocols vary tremendously from compiler to compiler. BDS C and Q/C have fairly straightforward function-calling protocols. Eco-C, on the other hand, has a tortuous function-calling sequence. And even though Q/C and Eco-C use the Microsoft assembler and linker, the code files produced by them are not compatible."

The problem that Mr. Clark mentions is also present in the 8086 (MS-DOS, CP/M-86) world. Code from different compilers cannot be mixed because:

1. Each requires its own main function.
2. A wide variety of link formats exist.
3. Calling conventions differ between compilers.

The first two points are essentially insurmountable problems from the end user's point of view. However, the third point can be overcome by adding dummy routines to convert calling conventions.

Mr. Clark makes some additional points concerning the deficiencies of the 8080 Microsoft .REL (relocatable) format. These are of interest since many 8080 C compilers rely on this format. His comments are summarized in Table I (below).

It is obvious that an enhanced standard is necessary for the CP/M-80 world. Even under MS-DOS, where Microsoft has enhanced the linkage editor format, problems still exist.

Microsoft's MS-DOS linkage format supports long symbols (31 unique characters) and has case sensitivity. Thus, it overcomes the objections that Mr. Clark posed for the 8080 .REL system. However, one nagging problem still remains: the dichotomy between object modules and libraries. Because of this dichotomy, the MS-DOS linker always includes the full contents of an .OBJ module during linking.

However, libraries are searched. To make a library efficient, each function must be compiled into its own .OBJ file. Each .OBJ file then becomes a single sub-block in the library, and all the functions of that sub-block file are included at program linkage. If libraries and object modules were equivalent, this problem would be overcome since the functions of an object module would be separable during linking.

Runtime Libraries

Nonstandard runtime libraries are a plague to programmers. They inhibit portability and introduce bugs when software is transported between different compilers. Charles Brady of New South Wales, Australia, writes:

"An enormous contribution to standardization of C programs would be the publication of a standard I/O library for BDS C, with, if necessary, a modified runtime package. The very fast and efficient compiler, particularly with the symbolic debugger tool, provides a very inviting environment for software development. It is a great pity that this means abundant nonstandard C. As there is no inherent reason why this should be the case . . . someone should be able to . . . produce a Unix compatible I/O library."

This point is especially well taken in view of the large amount of BDS C software available through the C User's Group of Yates Center, Kansas.

In the case of the runtime library itself, a clear standard exists. This standard is spelled out in *The C Programming Language*. All compilers have to do is support a proper subset appropriate to the environment in which they work.

Conclusion

In this column, I have included the follow-up discussion on Unix and on link formats/runtime libraries. The discussion has been mainly of a critical nature. However, this is not intended to be a condemnation of the systems but rather an impetus for readers to suggest new ideas to improve what we already have. Unix is a worthwhile standard and should be supported. C is a valuable tool but can be improved. This column is a forum where we can discuss, develop, and nurture new ideas about C and Unix. **DDJ**

Table 1.

Deficiencies in the Microsoft 8080 .REL Format and Related Software

1. M80 and DRI's RMAC assemblers support only six unique characters (all upper case). This is awkward for many purposes.
2. While the .REL format apparently will handle seven unique characters, neither M80 nor RMAC supports this.
3. Apparently the .REL format, M80, and L80 were designed to work with the Fortran-80 compiler, which permits only six character symbols. This is an old standard and does not reflect the needs of today's compilers. The absence of case sensitivity in symbols is especially limiting.

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 197.**

telecommunications? use SOFTCOM®

SOFTCOM is a smart terminal and file transfer utility all in one easy to use package. SOFTCOM can:

- make your system a terminal to a HOST time sharing system.
- capture data onto your disk from a HOST time sharing system.
- send text and program files from your disk to almost any type of computer.
- exchange any type of file with another SOFTCOM system with error detection and automatic retry.

SOFTCOM runs on all CP/M based computers. Three license options are available: single CPU ($150), dual CPU ($250) and multi CPU ($450). Call our software HOT LINE 906/228-7622.

The Software Store
706 Chippewa Square • Marquette MI 49855

Circle **no. 56** on reader service card.

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)

DYNAMIC RAM		
256K	150 ns	$85.00
64K	200 ns	5.97
64K	150 ns	6.09
64K	120 ns	6.90
16K	200 ns	1.56
EPROM		
27128	300 ns	$22.50
2764	250 ns	8.30
2732	450 ns	4.75
2716	450 ns	3.60
2532	450 ns	4.75
STATIC RAM		
6264P-15	150 ns	$35.97
6264LP-15	150 ns	42.00
6116P-3	150 ns	5.75

MasterCard VISA or UPS CASH COD
Factory New, Prime Parts µP∞
MICROPROCESSORS UNLIMITED
24,000 South Peoria Ave.
BEGGS, OK. 74421 **(918) 267-4961**

Prices shown above are for Jan. 13, 1984

Please call for current & volume prices. Prices subject to change. Please expect higher prices on some parts due to world wide shortages. Shipping and insurance extra. Cash discount prices shown. Federal Express Standard Air @ $5.99! Orders received by 6 PM CST can be delivered to you by the next morning.

Circle **no. 35** on reader service card.

Get the power of your Z80, and the elegance of direct access to CP/M functions from your high level programs with
SYNLIB *utility library*

SYNLIB *consists of MICROSOFT compatible object code that may be called from any high level language that uses MICROSOFT parameter passing conventions.*

SYNLIB *gives you extremely powerful array and buffer manipulation using the Z80 LDIR instruction; program access to the CP/M CCP console command line; high speed disk block I/O; a high quality random number generator; hex to ASCII conversion optimized by special Z80 instructions; program chaining; and more.*

And, because our programmer abhors a vacuum, each 8" floppy comes packed with some of the most valuable public domain software, including available source, absolutely free. You get **SWEEP,** *a menued disk utility that makes a computer phobe a systems programmer;* **UNSPOOL,** *so you can print* **and** *use your computer without buying an expensive buffer;* **/,** *to get multiple commands on a line;* **MODEM7,** *so that you too can join the free software movement; and many others.*

SYNLIB $50.00 8" SSSD CP/M format
SOURCE: $100.00
Licensing for commercial use available.
SYNTAX CONSTRUCTS, Inc.
14522 Hiram Clarke, Houston, Texas 77045
(713) 434-2098

CP/M is a registered trademark of Digital Research, Inc.
Microsoft is a registered trademark of Microsoft Corp.

Circle **no. 61** on reader service card.

FREHEAT

Passive Solar Simulation

The complete original mainframe nodal passive solar simulation program is now available for microcomputers, with complete source code!

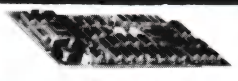
- 9 Overlays produce detailed Stats & Plots
- 3 Configurations: **Direct Gain** (Floor Mass)
  Mass Wall (with vents)
  Sunspace
- Supports: Aux Heating and Cooling, Thermo-circulation, Isothermal or Multi-layered primary mass, Secondary mass, Hourly TMY Weather input, Overhangs, Window Insulation, Day/Night Cycles, Heat pipes, selective surfaces, etc. **Only $175**

Why not choose the most sophisticated finite-difference nodal simulation for your microcomputer!

Western Wares
Box C
Norwood CO 81423
(303) 327-4898

Circle **no. 65** on reader service card.

80 CHARACTER VIDEO BOARD



S-100

Simpliway Products Co.

TYPE AHEAD KEYBOARD BUFFER
WORDSTAR/dBASE II OPTION
25 LINE NON-SCROLL OPTION
ADAPTABLE SOFTWARE
Z80 CPU & 8275 CRTC

VDB BARE BOARD FROM	$ 52.
PRE-SOLDERED (LESS ICs)	$191.
ASSEMBLED & TESTED FROM	$282.

P.O. Box 601
Hoffman Estates, Ill. 60195
(312) 359-7337

OEM & Dealer pricing available , VISA & M/C
$3.00 S&H, 5% cash discount, Ill. res. add 7% tax

dBASE is a trademark of Ashton-Tate Corp.
WORDSTAR is a trademark of Micropro Int'l Corp.

Circle **no. 54** on reader service card.

CGRAPH

DEVICE INDEPENDENT GRAPHICS SOFTWARE

- Vectors
- Character generator
- Windowing and clipping
- Re-entrant and re-interruptible
- Multiple simultaneous windows

Graphics programs independent of physical display, drives many displays at once. Completely documented in clear English with examples. Typical device drivers provided for hardware, including EPSON printers. Shipped as **C SOURCE CODE** on 8" SSSD CP/M or 5¼" MS-DOS disk. Specify standard (K&R) or BDS C version. Ask for availability for other operating systems or media. Send check for **$49.95** to

Systems Guild Inc.
P.O. Box 1085, Cambridge, MA 02142
617-451-8479

Circle **no. 62** on reader service card.

Advertise During

C Months

May - June - July

July Space Reservations Deadline:
May 3, '84
Material Deadline:
May 10, '84

Contact:
Walter Andrzejewski
Beatrice Blatteis
Alice Hinton

Dr. Dobb's Journal
P.O. Box E
Menlo Park, CA 94026
(415) 323 - 3111

Circle **no. 69** on reader service card.

JVS–FORTH $10
- SUPPORTS BYTE, INTEGER, AND REAL DATA
- COMPLETE ACCESS TO CP/M
- REQUIRES ASSEMBLER/LINKER BELOW

BASIC COMPILER $20
- SUPPORTS INTEGER, REAL, AND STRING DATA
- 25 STATEMENT TYPES
- GENERATES 8080 ASSEMBLER CODE
- RUN TIME LIBRARY SOURCE AVAILABLE ($50 EXTRA)
- MANUAL ONLY $5 (REF. WITH SOFTWARE PURCHASE)
- REQUIRES ASSEMBLER/LINKER BELOW

ASSEMBLER/LINKER $10
- SYMBOLIC 8080 RELOCATABLE ASSEMBLER
- LINK EDITOR (LIBRARY SEARCH CAPABILITIES)
- LIBRARIAN FOR LINK EDITOR LIBRARIES

FREE BROCHURE AVAILABLE

REQUIRES CP/M-80 V2.2 AND 32K
8" 3740 SSSD OR APPLE 5¼" 16-SECTOR
DISK FORMATS ONLY

WHEN ORDERING ADD $10 FOR
SHIPPING, HANDLING, AND MEDIA COSTS.
MASS. SHIPMENTS ADD 5% SALES TAX
MAKE CHECK OR MONEY ORDER PAYABLE TO

JV SOFTWARE
P. O. BOX 684•NEWTON, MA 02162

CP/M is a trademark of Digital Research, Inc.
APPLE is a trademark of Apple Computer, Inc.

Circle **no. 28** on reader service card.

CP/M 2.2 **INTRCEPT** CDOS

NEW version for 1984!

- Enables programs written for Digital Research CP/M 2.2 to run under Cromemco CDOS and vice versa. **INTRCEPT** release 3 automatically recognizes the host system, and emulates CP/M 2.2 if the host is CDOS, or emulates CDOS if the host is CP/M 2.2.
- No programming, delivered ready-to-run.
- Customizable...comes with CDOS emulator source, CP/M 2.2 emulator source optional.
- Z80 assembly language, no program or operating system modifications.

$150 w/CDOS emulator source
$250 w/CDOS & CP/M emulator source

8" SSSD, inquire about 5¼"
add $3 shipping, add 6% tax in CA
VISA, MC,check

PRO *microSystems*
16609 Sagewood Lane
Poway, California 92064
(619) 693-1022

Circle **no. 45** on reader service card.

OF INTEREST

by Michael Wiesenberg

TEXnical Information

If Stanford's Donald Knuth has his way, typesetting will no longer be an esoteric trade practiced only by a privileged and knowledgeable few. Computerized typesetting with its almost infinite flexibility permits individuals to set their own copy. No longer need the wasteful method be followed of one person producing copy on a word processor and printing that copy out on a daisy-wheel printer for another individual to reenter on the front end of a phototypesetter. That, of course, introduced new mistakes by the person doing the typesetting, an individual often untrained in the discipline addressed by the original author and thus unlikely to recognize many typographical errors.

Also the typesetting equipment, even if computerized, does not have error-checking facilities like the spelling and grammar checkers available on microcomputers. Sometimes the only interface between the supposedly computerized phototypesetting front end and the phototypesetter itself is not even a direct link but an antiquated paper tape. A completely new tape has to be punched each time a particular document is updated.

Some people predicted the death of the typesetting industry, or at least danger to members of the typesetting unions, when Knuth released **TEX** (pronounced *tek*) in 1978; but in fact what will happen is that typesetting will become more universally accessible, creating more jobs. When word processing was introduced, it did not – as some doomsayers had predicted – render secretaries obsolete; it just moved some of them from the typewriter to the terminal. What *did* happen was that more documents were produced. With the proliferation of computerized typesetting, we'll just see more good-looking documents.

Now, how can you get TEX onto your TRS-80 or Apple? Well, that's the bad news. You can't. Until recently, TEX was available only for mainframes. But here's the good news. You can get it on a system that has, depending on configuration, a minimum of 500K of main memory. Such systems range from the IBM PC to Hewlett-Packard computers. Since the TEX software is in the public domain, you could theoretically put it onto a virtual memory system with as little as 128K memory, but you'd have to adapt the software yourself and write the drivers for whatever output device you needed. That shouldn't be too hard for you Pascal wizards, right?

If you plan to implement TEX yourself and want the public domain software, you can get it from Maria Code by special arrangement with the Computer Science Department of Stanford University. 1200-foot tapes in various formats are $82 per tape if you supply the tape, or add $10 if you don't. The generic tape contains TEX82, WEB, fonts, and other miscellany; you need a Pascal compiler. **Reader Service No. 101.**

The TEXbook

Knuth's first version, TEX78, was written in SAIL. TEX82 is written in WEB, a system of structured documentation that combines the features of both a programming language (Pascal) and a document formatter (TEX). To illustrate Dr. Knuth's marvelous sense of humor (also found, by the

```
If the generator is now tuned to $f_0$, where the receiver power gain
is $G(f_0)$, the total power output will be
$$P=FkT\int^\infty_0 G(f)\, df+SG(f_0)$$
where $S$ is the available signal power. If $S$ is now adjusted so that
$P=2N$, i.e., so that the reading of the square law detector doubles, then
$$FkT\int^\infty_0 G(f)\, df=SG(f_0)$$
$$F={S\over kT{1\over G(f_0)}\int^\infty_0 G(f)\, df}={S\over kTB}
  \eqno{(3)}$$
where
$$B={1\over G(f_0)}\int^\infty_0 G(f)\, df=\hbox{effective noise bandwidth}$$
In order to find $B$ it is usually necessary to measure the power gain
as a function of frequency and integrate the curve graphically. This
method has major disadvantages, however, since it is time-consuming
and since $G(f)$ may vary with different conditions of adjustment.
A further disadvantage is the difficulty of determining $S$ with the required
accuracy at the low levels involved.
```

Figure 1.

Source code for the HPTEX output shown in Figure 2, below.

If the generator is now tuned to f_0, where the receiver power gain is $G(f_0)$, the total power output will be

$$P = FkT\int_0^\infty G(f)\,df + SG(f_0)$$

where S is the available signal power. If S is now adjusted so that $P = 2N$, i.e., so that the reading of the square law detector doubles, then

$$FkT\int_0^\infty G(f)\,df = SG(f_0)$$

$$F = \frac{S}{kT\frac{1}{G(f_0)}\int_0^\infty G(f)\,df} = \frac{S}{kTB} \tag{3}$$

where

$$B = \frac{1}{G(f_0)}\int_0^\infty G(f)\,df = \text{effective noise bandwidth}$$

In order to find B it is usually necessary to measure the power gain as a function of frequency and integrate the curve graphically. This method has major disadvantages, however, since it is time-consuming and since $G(f)$ may vary with different conditions of adjustment. A further disadvantage is the difficulty of determining S with the required accuracy at the low levels involved.

Figure 2.

Actual output of HP2688A Laser Printer using HPTEX.

Source and output courtesy of Boise Division of Hewlett-Packard.

way, in his *TEXbook*): you pass a .WEB source file through the TANGLE program to produce a .PAS Pascal file or through the WEAVE program to produce a .TEX file (that can now be processed by TEX). ("Oh what a tangled web we weave. . . .")

You cannot use TEX without a copy of *The TEXbook*. It is a delight to read, though really meant only for actual users of the software because it has a lot of exercises for you to try out. If you don't do them, you won't be able to follow what he's talking about. If you are familiar with his monumental series, *The Art of Computer Programming* (considered by most programmers to be the best thing written about programming), you have an idea of the level of expertise required for *The TEXbook*. Published jointly by Addison-Wesley and the American Mathematical Society, the book may be ordered from the latter for $15 (including p. and h., or add $2 for first class, $3 for airmail outside the U.S.). For 438 pages, most of it in nine- and ten-point type, that's a bargain! **Reader Service No. 103.**

HPTEX

The version I have been using is HPTEX, Hewlett-Packard's implementation of TEX82, on a 68000-based HP series 200 Pascal workstation, model 36. The software costs $4000. A typical configuration includes 9836A (the workstation with 500K RAM, Pascal, BASIC, UCSD editor, and several other utilities), $12,110, three more 256K memory boards at $1060 each, a disk interface for $605, DMA card for $500, 65Mb hard disk, $17,350, and the HP2688A desktop laser printer for $26,000. (You could save some money by substituting a model 26: you need 1.25Mb main memory, not necessarily the 1.75 I've been using; you can "get by" with 15Mb storage – although 65 is recommended – and you don't necessarily need the high-speed memory access, which also eliminates the need for the disk interface card.)

With this system, I can literally typeset an entire book by myself. I can choose from any of over 340 fonts, which come in sizes ranging from five to 40 points. A powerful macro set allows easy specification of any format. Dr. Knuth originally developed TEX so that complicated mathematical formulas in his books would be set exactly as he wanted, rather than as the typesetter felt it should look. TEX really shines at this, but it also excels at easily setting tables and other complicated formats.

HPTEX comes with an extensive set of macros that permit, among other things, multicolumn mode, with each column using a different font and a different measurement between lines, automatic generation of table of contents, and automatic indexing. I can define macros myself that permit juggling figures and tables around in a manual without worrying about changing the numbers (what happens to Figure 3-1, for example, and all the figures thereafter, when you have to insert two figures at the front of a manual?) of each. Many other computerized typesetting systems do not allow complete page layout, but this one does. Figures automatically float to a page with enough room for them, footnotes fall on the page on which they are referenced, and if material is added, the rest of the text automatically reformats itself.

Those familiar with typesetting know that any time anything is changed in a book, the effect ripples through the rest of the book, often requiring intervention by the typesetter on each page. Because things move around, technical writers have learned never to refer to other sections by page number; with TEX, this is no longer the case.

The heart of the HPTEX system is the HP2688A laser printer, which is no bigger than a computer terminal. The resolution produced by this little marvel is 300 dots per inch. While that does not approach the 3000 dots per inch of the Autologic APS-5, it is considerably better than most dot matrix or daisy-wheel printers. And quieter. And faster. **Reader Service No. 105.**

And For The 1000

Third-party vendors provide TEX for other HP computers, including the 3000 and the 1000. On the 1000 minis and micros with RTE-6/VM or RTE-A operating systems, JDJ Wordware offers **TEX/1000**. You need at least 512K main memory and 2Mb disk storage. Output devices include the Toshiba P1350 and Epson MX80, with HP2688A soon to be supported. TEX/1000 formats pages at 5 to 30 seconds each, depending on processor and memory. The output driver builds a page in memory, and then dumps the graphics to a printer. $2500. **Reader Service No. 107.**

Tick Tock

TYX Corporation offers an implementation of TEX, written in C rather than the usual Pascal (or SAIL), on DEC Rainbow, Victor 9000, IBM PC, and their own integrated **TyXSet 1000** system. The latter consists of a minicomputer that supports up to 24 users, a Canon Laser printer, a Mergenthaler Omnitech laser typesetter, terminals, and a typesetting interface. In fact, this system could provide the entire front end for a typesetting system suitable for a magazine or the publications department of a medium-sized company. No longer would a magazine have to wait several days to look at galleys and several more for the corrections.

TYX also offers smaller versions of their system configured on DEC Rainbow, Victor 9000, or IBM PC. All come with software and a friendlier interface than normally comes with TEX. Various function keys insert commands as needed, including the right number of beginning and ending group delimiters. All systems include a screen preview mode. While this implementation is TEX78, TYX expects to have TEX82 within a year. TYX also provides the service of taking TEX output files from customers who do not have phototypesetters and providing camera-ready copy from their own Autologic. **Reader Service No. 109.**

Get on the TUG

If you plan to use TEX, or attempt to implement it on your system, you will want to talk to others who have already done it. TEX has been put onto everything from Z80 systems to IBM 370s. The **TEX Users' Group** puts people in touch with "site coordinators," those in your area who have already implemented TEX with a system like yours. They also publish *TUGboat*, have yearly meetings, sell TEX tee shirts, etc. They'll send you an information packet containing *TUGboat* 1 #1, "TEX and Metafont: Errata and Changes," the TEX Users' Group Membership List (with the over 800 members sorted by name, computer, and output device), and forms for ordering TEX82 and joining TUG. You can also get information on video-taped instruction and special in-house courses on TEX. **Reader Service No. 111.**

Lost of Storage

Optical disk storage may be available within a few years for most micros, and at the prices now being paid for hard disks. The advantage is storage not in megabytes, but in *gigabytes*! Companies like Optimem, Thomson CSF, Panasonic, and 3M are working in this field. Who needs that kind of storage, you ask? Hospitals, storing thousands of digitized x-rays for immediate display, each of which can require 512 by 512 pixels by 16 bits deep, CAD systems, and movie production companies, who need to store millions of frames with virtually instantaneous recall. Did you know that credit cards are *now* available that can hold megabytes of information? And here's an ominous development: the military is very interested in optical drives because they feel such drives to be the only medium that could survive a nuclear attack. You can read all about all of this in *Optical Memory News*, $295 per year for six issues. **Reader Service No. 117.**

Improving the Compatibles

The **STM Personal Computer**, from STM Electronics (the folks who brought you the Pied Piper portable), a 17-pound, IBM PC-compatible, transportable computer based on the 80186 true 16-bit CPU, with integrated 16-line, 84-character, LCD, backlit by an electroluminescent panel for total darkness use, 40-column thermal printer, 94-key (including numeric pad and 10 function keys) keyboard, programmable 300- to 1200-baud auto dial/auto answer direct connect data communications modem, 256K RAM (expandable to 512), dual 1Mb (800K each formatted) floppies, standard video monitor support with windowing software and high resolution graphics, parallel port, two programmable serial ports, hard disk interface, edge connector to send I/O to an external expansion box, and integrated software, running under MS-DOS 2.0, with word processing, spreadsheet, database, graphics functions and communications software all standard, costs $3000, with a desktop version for $2500. **Reader Service No. 115.**

Cross Assembling 68K

Quelo wants us to know that their cross assembler for the MC68000 runs not only on CP/M-80, but also on CP/M-86, CP/M-68K, and PC-DOS, and they offer portable C source code. You get up to 31 characters in symbols, lowercase distinction, byte object linking, XOR, MOD, and NOT, *break* and *elseif* for control structures, error messages in English, a linker with limitless program size and number of modules that can be linked, and linker resolution of complex expressions. **Reader Service No. 119.**

Light Up Your Screen

Tech-Sketch Light Pens, for Commodore, Apple, and Atari, control the cursor or select menu options without hardware modifications. The **LP10-S** requires screen contact, while the **LP15-S** is a high-resolution pen that operates up to six inches from the screen. From $39.95. **Reader Service No. 113.**

DDJ

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service **No. 198.**

CDE SOFTWARE

CHECKS AND BALANCES $49.95
At last! A *full-screen* editing checking and finance program so informative and easy to use you'll wonder how you lived without it! Requires Z-80 processor and CP/M 2.2 or compatible system.

ORIGINAL ADVENTURE $29.95
CDE offers the fastest running version of this classic computer game available for a microcomputer.

CATCHUM and LADDER $39.95
Now you can enjoy the excitement of arcade games on your CP/M computer with these two variations of popular screen games.

DISKS from Allenbach. Lifetime certification and hub-protector rings.

	Single Sided Single Density	Single Sided Double Density	Double Sided Double Density
5¼"*	--	$27.95	$32.95
8"	$30.95	$35.95	$42.95

Visa and MasterCard accepted. Write or phone for our complete catalogue of games, business and development software, and accessories for your microcomputer. Our telephone—(213) 661-2031.

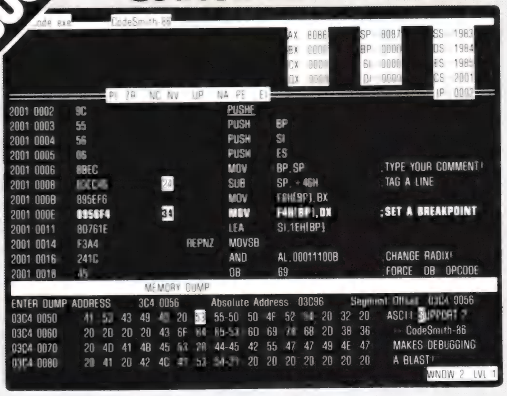
2463 McCready Avenue • Los Angeles, CA 90039

Circle **no. 7** on reader service card.

NEW IBM-PC DEBUGGER

"...by far the best debugging tool available for the IBM-PC today."
S. Lerner, *Multi Systems Group*

CodeSmith™-86


```
Code exe      CodeSmith-86
                                      AX 8086   SP 8087   SS 1983
                                      BX 0000   BP 0000   DS 1984
                                      CX 0000   SI 0000   ES 1985
                                      DX 0000   DI 0000   CS 2001
          PL ZR  NC NV  UP  NA PE  EI                     IP 0002
2001 0002  9C                 PUSHF
2001 0003  55                 PUSH    BP
2001 0004  56                 PUSH    SI
2001 0005  06                 PUSH    ES
2001 0006  8BEC               MOV     BP,SP          ;TYPE YOUR COMMENT!
2001 0008  83EC46      24     SUB     SP,+46H        ;TAG A LINE
2001 000B  895EF6             MOV     F6H[BP],BX
2001 000E  8956F4      34     MOV     F4H[BP],DX     ;SET A BREAKPOINT
2001 0011  8D761E             LEA     SI,1EH[BP]
2001 0014  F3A4         REPNZ MOVSB
2001 0016  241C               AND     AL,00011100B   ;CHANGE RADIX!
2001 0018  45                 DB      69             ;FORCE DB OPCODE
                         MEMORY DUMP
ENTER DUMP ADDRESS     3C4 0056    Absolute Address 03C96   Segment Offset 03C4 0056
03C4 0050    41 53 43 49 49 20 53 55-50 50 4F 52 54 20 32 20   ASCII SUPPORT 2
03C4 0060    20 20 20 20 43 6F 64 65-53 6D 69 74 68 2D 38 36       CodeSmith-86
03C4 0070    20 4D 41 4B 45 53 20 44-45 42 55 47 47 49 4E 47   MAKES DEBUGGING
03C4 0080    20 41 20 42 4C 41 53 54-21 20 20 20 20 20 20 20   A BLAST!
                                                                    WNDW 2 LVL 1
```

- Simple/single keystroke commands
- Scroll up/down thru full-screen disassemblies
- SCREENSAVE Mode
  Saves and restores user's graphic display when breakpoint hit
- Continuous monitoring of selected memory
- Hundreds of tagged breakpoints supported
- Disassemble selected ranges of memory code to disk—compatible w/IBM Assembler
- Many other inspired features
- Requires MS-DOS & 160K RAM

MasterCard • VISA • COD Orders

VISUAL AGE

642 N. Larchmont Blvd., Los Angeles, CA 90004
(213) 439-2414

CodeSmith, TM International Arrangements, Inc.
MS, TM Microsoft Corp. IBM, TM International Business Machines Corp.

INTRODUCTORY OFFER ONLY
Calif. residents add 6.5% sales tax
$145.00

Circle **no. 64** on reader service card.

MEGA-BYTE SPECIALIZING IN HARD DISKS AND HIGH PERFORMANCE STREAMING TAPE UNITS

MD - 10 11 MEGABYTE (FORMATTED) 5 1/4" HARD DISK Including **POWER! $1995**

POWER!

Check out InfoWorld's Rave review of POWER Nov. 8/82)

55 CP/M UTILITIES IN ONE PACKAGE

...The super program that puts you in control of CP/M (or MS-DOS) and your winchester.

POWER automatically numbers disk files. Just pick file number to Copy, Erase, Reclaim, Type, etc.
...Your computer feeds the file names automatically. You do NO typing! NO typing error ever!
YOU DON'T NEED SYSTEM DISK IN ANY DRIVE.
No more BDOS ERROR!

YOU Test and Fix bad disks! Reclaim accidently erased files or programs! Single step thru memory up or down! Search, view, change memory or disk in a snap! See Status and File Size instantly! Verify check-sums for programs! Load or Save programs at any address.

55 prompted user-friendly funstions for housekeeping and a 120 page easy-read user's guide make POWER your most often used software. You'll use it every day!

Now, POWER! also includes a special program that lets you lock sensitive files, so that only you can access then. Without the secret PASSWORD which you can create and change at will, no prying eyes will ever know your secret file even exists.

SOME MAJOR POWER COMMANDS:

REN	CM	WRITEGR	DISK	STAT
TYPEA	USR 2	DUMPA	SPEED	SETDIR
SIZE	ERA	SEARCH	WRITE	RECLAIM
XUSER	TYPEX	?	DUMPX	DS
SETRO	CHECK	COPY	FILL	READ
GROUP	TEST	TYPEH	EX	DUMP
SAVE	SETWR	EXIT	DIR	MOVE
READGR	LOG	USER	TYPE	JP
DUMPH	LOAD	SETSYS	RUN	

For this low price your winchester will be delivered completely assembled and tested, with drive controller, case, power supply, cabling, Z-80 interface and the best disk controller software on the market.

- The unique and simple interfacing system does not tie up existing ports.
- Significantly faster than other winchester subsystems which interface through the IEEE-488 port.
- CP/M drivers require minimum memory overhead (about 2K-other systems require as much as 6k)
- The MD-10 can read or write a 64K file in less than four (4) seconds.
- A network system will be available shortly which can support up to sixteen mixed types of computers from one MD-10 or larger disk subsystem.
- With POWER! Software files can be code word protected.
- Other systems available: 22 and 44 megabytes

MD-20 with 22MB formatted: $2895

MD-44 with 44MB formatted: $3995

- Up to eight (8) winchester subsystems can be interfaced to one computer.
- Software supports 32 different user areas per MD-10.
- Backing up hard disk files is simple with the special software which is provided with all subsystems.
- With a single hard disk installation, the MD-10 subsystems becomes units A and B with the standard drives being designed E and F if a second MD-10 is installed later it becomes units C and D.
- MD-10 or larger systems will interface with IBM PC or any Z-80 computer (CCS, APPLE (CP/M), ZENITH/HEATH, NORTHSTAR, GODBOUT, XEROX 820, Z-80/S100, ALSPA, or TRS-80 MOD II) using CP/M, OASIS, PCDOS 2.0, and Godbout Software supports both CP/M and MP/M816.
- Double density modifications are available allowing you to later increase the capability of an MD-10 to about 20 megabytes, MD-20 to 40, etc.
- **Full One-Year Warranty**

HIGH PERFORMANCE STREAMING TAPE

Unique! Simple! Compatible!

- **Field-proven Archive Sidewinder tape unit.**
- **Interfaces with ANY SASI/SCSI compatible hard disk peripheral.**
- **Full CRC error-checking**
- **Superb reliablity.**
- **Full One-Year Warranty.**
- **Fast (up to 20MB in less than 5 minutes, up to 40MB in less than 10 minutes).**

$1795.00 20MB
$1995.00 45/60MB

$29.95 3M DC300XL
Streaming Tape Cartridges

MEGA-BYTE

ORDER

Name	Qty.	Price	Total
VISA ☐ M/C ☐ COD ☐		Total Order	
Card # ________ Exp. Date ______		If Utah Tax	
Signature ________		Total Enclosed	

TO ORDER OR FOR MORE INFORMATION:
(801) 257-7033 or mail to MEGA-BYTE INC. RT. 3 Box 35, Tremonton, Utah 84337
TERMS: Cashiers Check, VISA, M/C or COD.
Shipping charges added to all orders.

MasterCard VISA

NAME ____________________
COMPANY ____________ TITLE ________
ADDRESS ____________ CITY ________
STATE ____________ ZIP ________
TEL ( ) ____________

CP/M is a registered trademark of Digital Research. OASIS is a proprietary product of Phase One Systems Inc. Z-80 is a trademark of Zilog. MD-10 is a registered trademark of Media Distributing.

Circle no. 33 on reader service card.

ADVERTISER INDEX

Reader Service No.	Advertiser	Page No.
1	2500 AD Software	52,53
2	Ashton-Tate	2
3	Avocet Systems, Inc.	29
4	BD Software	71
5	BG Micro	7
6	Bonnie Blue Software	44
70	Borland International	104
7	California Digital Engineering	100
8	Computer Design Labs	21
9	Computer Friends	95
10	Computer Innovations	35
11	Computer Resources of Waimea	59
12	Creative Solutions	75
13	C Users' Group	84
14	C Ware	91
69	*DDJ* Advertise	97
63	*DDJ* Back Issues	47
15	Data Access Corporation	103
16	Datalight	55
17	Data Management Associates, Inc.	13
18	Dedicated Micro Systems, Inc.	65
19	D & W Digital	73
20	Ecosoft, Inc.	67
21	Foehn Consulting Engineering	84
22	GGM Systems	65
23	GTEK, Inc.	43
24	Hallock Systems Consultants	95
25	Harvard Softworks	66
26	Hawkeye Grafix	62
27	Integral Quality	68
28	JV Software	97
29	Laboratory Microsystems	3
30	Lifeboat Associates	33
31	Loki Engineering	39
32	Manx Software	7
33	Mega-Byte	101
34	MicroMotion	30
35	Micro Processors Unlimited	97
36	Miller Micro Computer Services	26
37	Mountain View Press	9
38	Mu Software	55
39	Mylstar Electronics, Inc.	41
40	Novum Organum	93
41	Overbeek Enterprises	71
42	Pascal & Associates	49
43	The Programmer's Shop	51
44	PRO Microsystems	59
45	PRO Microsystems	97
46	Puterparts	102
47	Quelo	102
48	Quest Research	47
49	Quic-n-Easi Products, Inc.	11
50	Rational Systems	89
*	Edward Ream	31
51	Revasco	89
*	Rocky Mountain Software Systems	19
52	SemiDisk Systems	25
53	Shaw Laboratories	69
54	Simpliway Products Co.	97
55	SLR Systems	23
56	The Software Store	97
57	Software Writers Intn'l Guild	17
58	Software Writers Intn'l Guild	85
59	Solution Technology, Inc.	81
60	Solution Technology, Inc.	79
61	Syntax Constructs, Inc.	97
62	Systems Guild, Inc.	97
64	Visual Age	100
65	Western Wares	97
*	Whitesmith, Ltd.	4
66	John Wiley & Sons	43
67	Mark Williams & Co.	15
68	Workman & Associates	93

Available now.

The Best of Both Worlds

High-performance

CompuPro Hardware

with high-performance

Cromemco Software.

Cromemco's MC68000-Z80 CROMIX multi-tasking operating system with drivers to support CompuPro hardware. (requires Cromemco DPU)

Minimum configuration:

DPU - SystemSupport1 - Disk1 - Interfacer3 or 4 - 192K RAM

CROMIX with drivers to support minimum configuration $890.
Special drivers only $295.

The following products can be added to any 8 or 8-16 bit CROMIX system:

SCSI hard disk drivers $195.
15Mb formatted hard disk subsystem $2095.
30Mb (two drives) $2995.

MDrive-H drivers $195.
CompuPro 512K MDrive-H hardware $1695.

Other drivers in development... custom inquiries welcome.

Authorized CompuPro Dealer.

PuterParts®

2004 4th Avenue
P.O. Box 12339
Seattle, WA 98111-4339

Telephone (206) 682-2590

OPEN
2-6 Tuesday-Saturday

"PuterParts" is a registered trademark of Katharos Companies
Additional trademarks: Z80, Zilog: MC68000, Motorola: CROMIX, DPU, Cromemco: SystemSupport1, Disk1, Interfacer3, -4, MDrive-H, CompuPro.

Circle **no. 46** on reader service card.

PORTABLE

68000

ASSEMBLER

MOTOROLA SYNTAX **MACROS**
LINKER **CONDITIONALS**
OBJECT LIBRARIAN
CROSS REFERENCE
STRUCTURED PROGRAMMING FEATURES

OBJECT CODE OPTIMIZED FOR CP/M-80, CP/M-68K OR IBM-PC | **$300**

PORTABLE C SOURCE | **$750**

Quelo
843 NW 54th
Seattle, WA 98107

(206) 784-8018
mornings
Dick Curtiss

CP/M is a trademark of Digital Research
IBM-PC is a trademark of IBM

Circle **no. 47** on reader service card.

Like the molecules in a snowflake, the elements of a computer database can be structured and related in an infinite variety of combinations. Being able to present these combinations quickly and efficiently, with maximum flexibility and minimum programming knowledge, is the mark of excellence which sets the sophisticated database management system apart from the ordinary.

Thousands of satisfied users, from amateur and professional programmers to government agencies, major corporations and industries, have found that DataFlex has no equal in applications development software.

Write or phone for our latest literature and a list of existing applications.

Compatible with CP/M, CP/M-86, MP/M-86, MSDOS, PCDOS, TurboDOS, Novell Sharenet, PC-Net, Molecular N-Star, Televideo MmmOST, Action DPC/OS, Omninet, IBM PC w/Corvus and OSM Muse.

DATAFLEX

APPLICATIONS DEVELOPMENT SOFTWARE

DATA ACCESS CORPORATION

8525 S.W. 129th Terrace, Miami, Florida 33156 (305) 238-0012 TELEX 469021 Data Access CI

MSDOS is a trademark of Microsoft. CP/M and MP/M are trademarks of Digital Research. DataFlex is a trademark of Data Access Corp.

Circle no. 15 on reader service card.

ENTER . . . THE FUTURE!

Introductory offer

$49.95

THIS IS THE PASCAL COMPILER EVERYBODY'S BEEN WAITING FOR... EVERYBODY EXCEPT THE COMPETITION!

Extended Pascal for your IBM PC, APPLE CP/M, MS DOS, CP/M 86, CCP/M 86 or CP/M 80 computer features:

- Full screen interactive editor providing a complete menu driven program development environment.
- 11 significant digits in floating point arithmetic.
- Built-in transcendental functions.
- Dynamic strings with full set of string handling features.
- Program chaining with common variables.
- Random access data files.
- Full support of operating system facilities.
- And much more.

ORDER YOUR COPY OF **TURBO PASCAL** TODAY TO TAKE ADVANTAGE OF OUR INTRODUCTORY SPECIAL.

For Visa and MasterCard orders call toll free

1-800-227-2400 X 968

IN CA: 1-800-772-2666 X 968

(lines open 24 hrs. a day, 7 days a week)

Dealer & Distributor Inquiries welcome.

	Turbo Pascal	IBM Pascal	Pascal MT+
PRICE	49.95	300.00	595.00
Compile & Link speed	1 second!!!	97 seconds	90 seconds
Execution speed	2.2 seconds	9 seconds	3 seconds
Disk Space 16 bit	33K w editor!	300K + editor	225K + editor
8 bit	28K w editor!	Not Available	168K + editor
8 and 16 bit	YES	NO	YES
built-in editor	YES	NO	NO
Generate object code	YES	YES	YES
One pass native code compiler	YES	NO	NO
Locates RunTime errors directly in source code	YES	NO	NO

Benchmark data based on EightQueens in "Algorithms + Data Structures = Programs" by N. Wirth, run on an IBM PC.
Turbo Pascal is a trademark of Borland International. MT+ is a trademark of MT MicroSystems. IBM is a trademark of International Business Machines.

LEARN TO WRITE A SPREADSHEET

Our Introductory offer includes MICROCALC, a spreadsheet written in Turbo Pascal. It will be on your disk, and ready to run. And we've included the source code to show you exactly how a spreadsheet is written!

[redacted]

Turbo Pascal includes a 250 page bound manual with extensive explanations and many illustrative examples.

Turbo Pascal $49.95 + $5.00 shipping per copy.

Check _____ Money Order _____
VISA _____ Master Card _____

Card #: _____
Exp date: _____ Shipped UPS

BORLAND INTERNATIONAL

Borland International
4807 Scotts Valley Drive
Scotts Valley, California 95066

My system is: 8 bit _____ 16 bit _____
Operating system: CP/M 80 _____
CP/M 86 _____ MS DOS _____ PC DOS _____
Computer: _____ Disk Format: _____
Please be sure model number and format are correct.

NAME: _____
ADDRESS: _____
CITY/STATE/ZIP: _____
TELEPHONE: _____

California residents add 6½% sales tax. Outside North America add $15.00. Checks must be on a U.S. bank, and in U.S. dollars. Sorry, no C.O.D.

Circle **no. 70** on reader service card.